Smart Squad Selection:

Enhancing Sri Lankan Cricket Team Composition through Data-Driven Insights

- Final Year Thesis –

Project ID: 44

Student Number: 21UG0036

Student Name: Shamika Konara

FACULTY OF COMPUTING AND INFORMATION & TECHNOLOGY

SRI LANKA TECHNOLOGICAL CAMPUS

FINAL YEAR PROJECT THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

B.SC. HONS. IN DATA SCIENCE

April 2024

Supervisor: Dr. D. S. Deegalla

Dedication

This thesis is dedicated to our families, friends, and mentors who have supported us throughout this journey. Their unwavering encouragement, patience, and belief in our capabilities have been invaluable in the completion of this research.

We extend our heartfelt gratitude to our supervisor, Dr.D.S.Deegalla for their invaluable guidance and mentorship. Their insights and expertise have played a crucial role in shaping this project into a meaningful contribution to the field of sports analytics and artificial intelligence.

Lastly, we dedicate this work to all cricket enthusiasts and researchers striving to revolutionize the game through data-driven innovations. May this study serve as a stepping stone for future advancements in intelligent sports analytics.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my/our own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgments

Shamika Konara (21UG0036)

3/4/2025

W.B.M.T.R. Wijesooriya (21UG0785)

3/4/2025

Acknowledgments

We would like to express our sincere gratitude to our supervisor, Dr. D.S. Deegalla for their continuous support, guidance, and insightful feedback throughout the course of this research. Their expertise and encouragement have been invaluable in shaping the direction and success of this project.

We are also grateful to Sri Lanka Cricket (SLC) and various cricket analysts in ESPN, Papare for providing us with crucial data and insights that contributed significantly to the development of our Smart Squad Selection system.

Additionally, I extend my sincere appreciation to my project partner, Tharinda Wijesooriya, for his dedication, hard work, and collaboration in making this project a success. His contributions and commitment were crucial throughout this journey.

Special thanks go to our families and friends, whose unwavering support and motivation have helped us persevere through the challenges of this research journey. Their encouragement has been a constant source of strength.

Lastly, we acknowledge our fellow researchers and academic peers at SLTC Research University, whose discussions and feedback have provided valuable perspectives and improvements to our work. This research would not have been possible without the collective efforts and contributions of everyone mentioned above.

Abstract

Cricket squad selection is a critical and complex process that requires analyzing multiple factors such as player performance, fitness, and team composition. Traditional selection methods rely heavily on selectors' expertise, which may introduce biases and inefficiencies. This research aims to develop a data-driven approach for Sri Lankan T20I team selection, leveraging machine learning and optimization techniques to enhance squad composition.

The study integrates player performance data from ESPN (International), Papare (LPL), and fan votes from Google Forms, ensuring a comprehensive evaluation. Performance scores are computed using statistical measures and machine learning-based weighting models, and an optimized playing XI is generated using a genetic algorithm. Additionally, a machine learning model predicts the best team for specific venues, and another model determines a player's selection probability based on fan voting trends.

The final implementation is a web-based platform with four key sections displaying the best squad selection based on different criteria. Furthermore, an AI chatbot provides insights into Sri Lankan cricket, assisting users with real-time queries. The results demonstrate the effectiveness of data-driven selection methods, offering a more objective and analytical approach to squad selection. This research contributes to modernizing cricket selection processes by integrating machine learning and optimization techniques.

Table of Contents

List of Figures

List of Tables

Nomenclature

GA – Genetic Algorithm

KNN – K-Nearest Neighbors

XGBoost – Extreme Gradient Boosting

RAG – Retrieval-Augmented Generation

IBaC – International Batting Consistency

IBoC – International Bowling Consistency

LBaC – LPL Batting Consistency

LBoC – LPL Bowling Consistency

IAC – International Allrounder Consistency

MAE – Mean Absolute Error

MSE – Mean Squared Error

RMSE – Root Mean Squared Error

R^2 – Coefficient of Determination

FoQ – Frequency of Quality

ML – Machine Learning

SR – Strike Rate

Inns – Innings

Fan-Based Model – Model developed using fan vote data

VPM – Venue Prediction Model

FSPM – Fan Selection Probability Model

1. Introduction

1.1 Background

Cricket is the most popular sport in Sri Lanka, with T20 International (T20I) matches being a major aspect of the game. Selecting the best possible playing XI is a crucial factor in achieving success in international cricket. Traditionally, team selection has been based on selectors' expertise, player statistics, and recent performances. However, with advancements in data science and machine learning, data-driven approaches can significantly enhance the selection process.

Our project, Smart Squad Selection: Enhancing Sri Lankan Cricket Team Composition through Data-Driven Insights, aims to leverage data analytics and machine learning to recommend the best Sri Lankan T20I squad for 2025. The study integrates data from multiple sources, including ESPN (international matches), Papare (Lanka Premier League - LPL), and fan votes collected through Google Forms. By utilizing performance metrics, weighting mechanisms derived from machine learning models, and genetic algorithms, the system generates an optimal playing XI based on various parameters. Additionally, a web-based application is developed to provide user-friendly access to squad recommendations and an AI-powered chatbot for cricket-related queries.

1.2 The Problem

Traditional player selection methods in Sri Lankan cricket have primarily depended on historical reputation, expert opinions, and recent performances in local tournaments. While these approaches have been the standard for years, they often overlook emerging talents and fail to dynamically adapt to changing match conditions. One of the major challenges with this traditional system is the presence of subjective biases, where selectors may favor certain players based on personal preferences rather than data-driven insights. Additionally, detailed performance metrics are not systematically integrated into the selection process, leading to inconsistent decision-making.

Another significant limitation is the inability to dynamically adjust team compositions based on match scenarios, opposition strategies, and venue conditions. Without real-time data-driven

analysis, the selected squads may not always align with the tactical demands of a given match. These challenges emphasize the urgent need for a structured, unbiased selection model that incorporates performance scoring algorithms and machine learning techniques. By leveraging advanced data analytics, cricket squads can be optimized to maximize team performance, ensuring that selection decisions are based on objective, quantifiable metrics rather than subjective judgments.

1.3 The Proposed Solution

To address these challenges, We proposed a machine learning-based Smart Squad Selection The proposed system integrates multiple data sources to generate an optimal T20I playing XI for Sri Lanka in 2025, ensuring a data-driven and unbiased selection process. The system consists of several key components that work together to optimize team selection:

The Data Collection & Processing module gathers player statistics from ESPN (for international matches), Papare (for Lanka Premier League - LPL), and Google Forms (for fan votes). The collected data undergoes cleaning and formatting, ensuring accuracy and consistency before being used for further analysis.

A Performance Scoring System is employed to assess players based on their contributions in international and LPL matches. These scores are calculated using weighted metrics derived from machine learning models, considering factors such as runs scored, batting averages, wickets taken, and economy rates. Players selected by fans are ranked based on the frequency of their selection, providing an additional perspective in the selection process.

To optimize the squad, a Genetic Algorithm for Team Selection is implemented. This evolutionary algorithm evaluates composite performance scores and iteratively improves the squad composition by selecting the most balanced and high-performing team based on predefined criteria.

A Venue-Based Prediction Model is integrated into the system, using machine learning techniques to recommend the best playing XI based on the match location. This ensures that selection decisions consider the impact of venue-specific conditions, such as pitch behavior and weather factors, on player performance.

Additionally, a Fan-Based Selection Probability Model is introduced to estimate the likelihood of a player's selection based on trends observed in fan voting data. This allows the system to incorporate public sentiment while maintaining a data-driven approach.

Finally, the system features an interactive web application and an AI-powered chatbot. The web application presents the selected squads, enabling users to explore different selection criteria and visualize key performance metrics. The AI chatbot, built using FastAPI and LangChain, allows users to ask cricket-related queries, retrieve player insights, and understand squad selections based on historical and predictive data models.

1.4 Deliverables and Milestones

The proposed solution will be implemented in phases, with measurable milestones ensuring systematic progress. The first stage involves data collection and processing, where player statistics from international matches, LPL, and fan vote data are scraped, cleaned, and structured to maintain consistency and accuracy.

Following data preparation, the performance scoring system is developed. This phase includes computing weighted performance scores for players using machine learning-derived weight values, ensuring a data-driven approach to player evaluation. Once the performance scoring mechanism is in place, the genetic algorithm is implemented to optimize squad selection by developing and testing an algorithm that generates the best possible team composition[4].

The next phase focuses on machine learning model development, which includes building a venue-based team prediction model to recommend the best playing XI based on historical performance at specific venues. Additionally, a fan selection probability model is introduced to analyze public voting trends and estimate the likelihood of player selection based on fan preferences.

With the core algorithms and models in place, the system is then integrated into a web application. This stage involves designing and developing an interactive interface where users can explore different squad combinations, view insights, and interact with system-generated recommendations. Finally, an AI chatbot powered by a Retrieval-Augmented Generation (RAG) model is integrated into the platform, allowing users to query cricket-related

information, access squad recommendations, and retrieve historical player statistics in real time.

By following this structured development approach, the project ensures a robust, scalable, and intelligent system for optimizing squad selection in Sri Lankan T20I cricket.

Phase	Deliverables	Start Date	End Date	Duration
Data Collection & Preprocessing	Collect, clean, and structure cricket data.	Week 1	Week 4	4 Weeks
Performance Scoring System	Develop ML-based performance scoring.	Week 5	Week 8	4 Weeks
Genetic Algorithm Optimization	Implement squad selection algorithm.	Week 9	Week 12	4 Weeks
ML-Based Prediction Models	Venue-based and fan selection models.	Week 13	Week 16	4 Weeks
Web Application Development	Develop and deploy web interface.	Week 17	Week 20	4 Weeks
AI Chatbot Integration	Integrate and optimize chatbot responses.	Week 21	Week 24	4 Weeks

Table i: Milestones

2. Related Works

2.1 Introduction

Related work is an essential component of any research as it establishes a foundation by analyzing prior studies in the domain. For our project, "Smart Squad Selection: Enhancing Sri Lankan Cricket Team Composition through Data-Driven Insights," we explored various research papers focusing on cricket squad selection, performance analysis, and machine learning applications in sports. Understanding past methodologies helped us refine our approach, avoid redundancy, and improve upon existing models.

2.2 Existing Research in Cricket Team Selection

2.2.1 Player Fitness and Squad Selection using Intelligence System

A study on Player Fitness and Team Squad Selection using Intelligence System in Sri Lanka proposed an AI-driven approach for selecting cricket squads based on player fitness levels. This research emphasized the role of real-time data analytics and injury tracking in team selection. We incorporated this concept by integrating player consistency metrics into our performance scoring system, ensuring that players selected for the squad have maintained a stable performance across formats.

2.2.2 Machine Learning for Cricket Squad Selection

The paper Cricket Players Selection for National Team and Franchise League using Machine Learning Algorithms employed machine learning techniques like decision trees, support vector machines, and neural networks to predict optimal squad selections. Our research adopted a similar machine learning-based feature selection approach, where we trained models using MSE and R^2 scores to derive the most important features affecting player performance. This research helped us refine our model selection process, leading to the use of XGBoost for fan-based selection probability and K-Nearest Neighbors (KNN) for venue prediction.

2.2.3 Genetic Algorithm for Optimal Team Selection

The paper Optimal One Day International Cricket Team Selection by Genetic Algorithm introduced a genetic algorithm-based optimization strategy for selecting a balanced playing XI. Inspired by this, we developed a genetic algorithm to generate the best playing XI, not only for a general team but also for venue-specific teams. We modified the genetic algorithm's fitness function to accommodate multiple factors such as batting consistency, bowling performance, and all-rounder impact[3].

2.2.4 Machine Learning for Squad Selection in Franchise and International Cricket

The research Squad Selection for Cricket Team Using Machine Learning Algorithms compared different machine learning models for selecting players in both international and franchise leagues. This work highlighted how various leagues require different selection criteria, which influenced our approach of separating International and LPL performances in our performance scoring system. By leveraging data from ESPN (International), Papare (LPL), and Google Forms (Fan Votes), we ensured that our model accounted for different competition levels.

2.2.5 Survey on Machine Learning in Cricket Team Selection

The paper A Survey on Team Selection in Cricket Using Machine Learning provided a comparative analysis of various algorithms used in previous studies for player selection. This helped us evaluate Random Forest, Linear Regression, XGBoost, and Decision Trees before concluding that XGBoost performed best for fan selection probability, while KNN was most suitable for venue prediction.

2.3 Integration of Related Work into Our Research

By analyzing previous research in player selection and sports analytics, our study incorporates several key enhancements to improve accuracy, objectivity, and adaptability in squad selection. These enhancements build on existing methodologies while introducing novel elements to optimize decision-making.

The Performance Scoring System was developed by leveraging feature importance analysis from prior studies. Multiple machine learning models, including Random Forest Regression,

XGBoost, and Lasso Regression, were trained and evaluated based on Mean Squared Error (MSE) and R^2 scores. The best-performing model was then used to derive an optimized weighted performance equation, ensuring that player rankings accurately reflect their contributions in international and domestic leagues.

To refine team selection, a Genetic Algorithm for Squad Optimization was implemented, improving upon previous approaches by enhancing the fitness function. This allowed the algorithm to generate both a primary squad and venue-specific optimal squads, adapting to different match conditions and opponent strategies.

The Venue Prediction Model was introduced to recommend the most effective playing XI for specific venues. Using historical squad data and winning team combinations, a K-Nearest Neighbors (KNN)-based model was trained to assess venue suitability. This model enhances team selection by identifying the best-performing squad for a given match location, considering factors such as pitch conditions and climate.

To incorporate fan-based sentiment, a Fan-Based Selection Probability Model was developed. Inspired by prior research on the role of public opinion in sports analytics, fan votes were collected via Google Forms, and a favoritism score was calculated. Multiple machine learning models, including Random Forest, Linear Regression, XGBoost, and Decision Tree, were tested, and the best-performing model was used to predict the likelihood of a user-selected team being chosen.

A significant innovation in our research is the RAG AI Chatbot for Cricket Insights. Unlike previous studies, which primarily focused on structured data, we incorporated a Retrieval-Augmented Generation (RAG) chatbot, trained on Sri Lankan school, domestic, and tournament cricket data. This chatbot, powered by FastAPI and LangChain, provides users with real-time, contextual responses to cricket-related queries, offering insights that are not easily accessible through standard online searches

.2.4 Summary

Our research built upon previous studies while introducing novel aspects such as a unified squad selection framework that integrates performance metrics, fan votes, venue-specific predictions, and AI-driven insights. By refining existing methodologies and addressing prior

research limitations, our work advances cricket analytics and provides a data-driven approach to Sri Lankan cricket team selection

3. Methodology

3.1 System Design

Our Smart Squad Selection System follows a modular design, integrating multiple data sources and machine learning techniques to generate an optimal playing XI for Sri Lanka's T20I team. The system architecture is structured into several key components, starting with data collection and preprocessing. This phase involves gathering player statistics from ESPN, Papare, and Google Forms, ensuring that data is cleaned, formatted, and processed to remove inconsistencies and handle missing values.

The performance scoring system then assigns weighted performance scores to players based on key batting, bowling, and all-rounder metrics. These weight values are derived from machine learning models, enabling a data-driven approach to evaluating player capabilities. To optimize squad selection, a genetic algorithm is applied to evaluate different team combinations based on computed performance scores, ensuring that the best-balanced squad is selected.

A venue-based prediction model is incorporated to recommend the most suitable playing XI for specific match locations by leveraging historical performance trends at different venues. Alongside this, the fan-based selection probability model analyzes fan voting trends to estimate the likelihood of a player's selection, ensuring that public opinion is also considered in the selection process.

All these components are integrated into an interactive web application, which includes an AI-powered chatbot. This chatbot, built using a Retrieval-Augmented Generation (RAG) model, allows users to access real-time cricket insights, team recommendations, and historical player performance data through natural language queries.

My role in this project was crucial in several areas. I was responsible for data collection and processing, which involved scraping, cleaning, and structuring player performance data from ESPN, Papare, and Google Forms to ensure accuracy and consistency in the dataset. I also developed the performance scoring system by implementing machine learning-derived weight values to create an objective and data-driven player evaluation mechanism. Additionally, I played a key role in the implementation of the genetic algorithm, developing and testing the optimization model to generate the best squad based on composite performance metrics. Another major contribution was the integration of the AI chatbot, where I incorporated a

Retrieval-Augmented Generation (RAG)-based chatbot to provide users with real-time cricket insights and assist in squad selection decisions.

Through these responsibilities, I ensured that our project effectively leveraged machine learning, optimization techniques, and AI-driven insights to enhance the squad selection process, making it more systematic, unbiased, and performance-driven.

System Architecture Diagram

First, let's discuss the system architecture design of our Smart Squad Selection system. The architecture was structured to ensure an efficient data-driven approach for selecting the optimal Sri Lankan T20I squad. Initially, data was gathered from multiple sources, including ESPN for international cricket statistics, the Papare website for Lanka Premier League (LPL) data, Google Forms for fan feedback, and additional school cricket data for the RAG chatbot. Web scrapers were then implemented to extract relevant information from cricket websites, followed by a data transformation process to standardize formats and ensure consistency. Fan selection data was also processed separately to analyze user preferences effectively. Once the data was structured, different performance analysis techniques were applied, where separate scoring algorithms were developed to evaluate international and LPL performances. Fan selection frequency was also analyzed, and a combined performance score was generated to integrate insights from all three sources. The core team selection process was handled by a genetic algorithm that identified the most optimal player lineup, while a venue-based machine learning model was used to refine the team selection for different match locations. Additionally, a fan selection prediction model was implemented to estimate the probability of players being selected based on user feedback. The processed results were then passed to the backend, which managed data storage and algorithm execution, ensuring smooth communication with the database. The web application frontend was designed with four key tabs—Main, International, LPL, and Fan—each displaying relevant squad selections. Lastly, a RAG chatbot was integrated to provide school cricket-related insights. This structured flow of data, from collection and processing to analysis and final team selection, enabled the system to provide an objective and data-driven approach to squad selection.

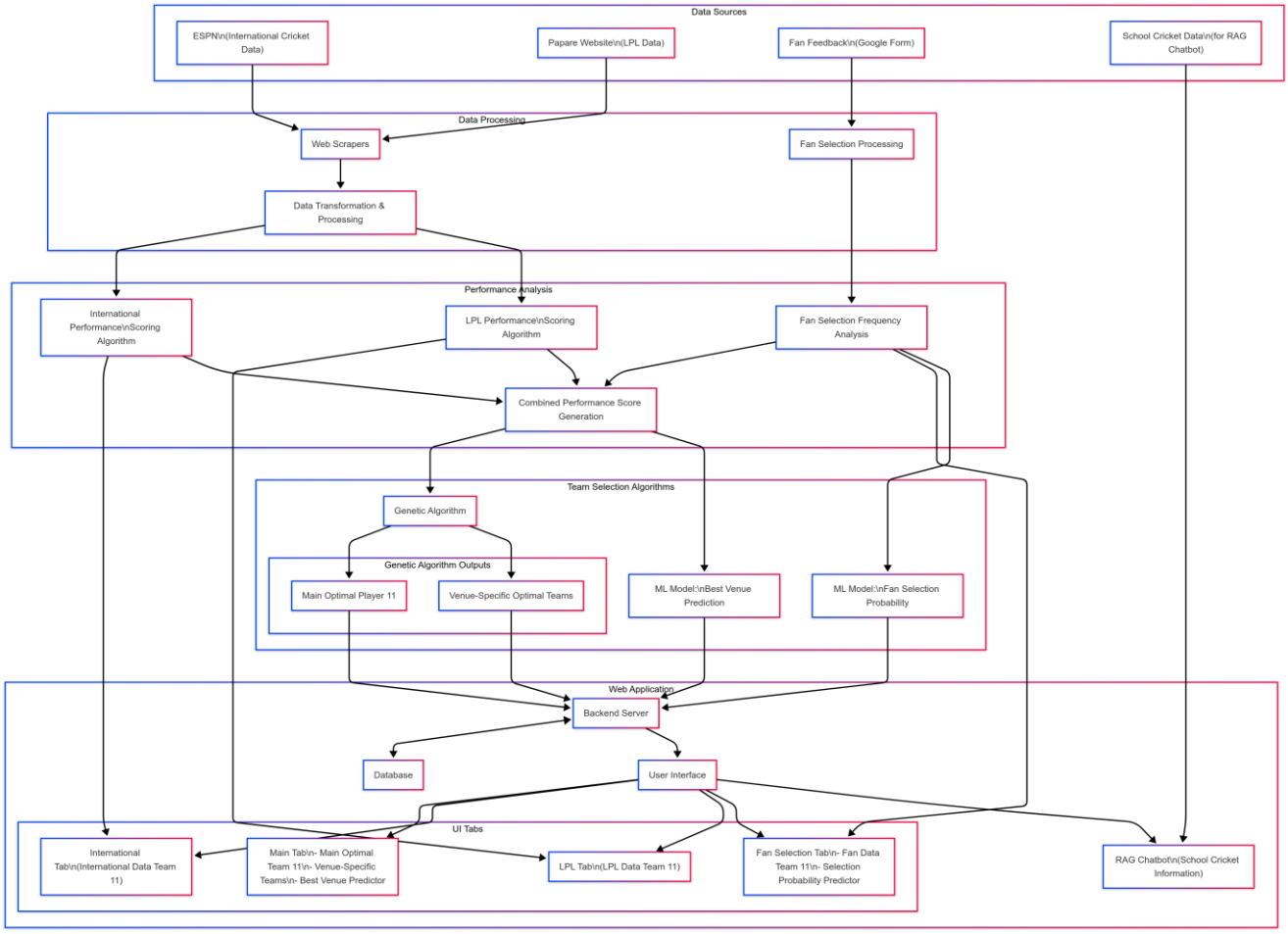


Figure i: System Architecture Diagram

3.1.1 Data Collection & Preprocessing

The system gathers cricket data from multiple sources to ensure a comprehensive and data-driven approach to player selection. Data is extracted from ESPN for international matches, Cricbuzz for additional match statistics, Papare for Lanka Premier League (LPL) performances, and Google Forms to capture fan votes. This diverse dataset enables the system to assess players based on a combination of professional match performance and public sentiment.

Once collected, the data undergoes preprocessing to enhance its quality and usability. Data cleaning is performed to remove duplicates, inconsistencies, and irrelevant entries. Formatting ensures uniformity in numerical values and categorical attributes, making the dataset suitable for analysis. Handling missing values is a crucial step, where missing records are either imputed or removed based on statistical relevance. Lastly, feature engineering is applied to extract key attributes, such as venue-based performance, recent form, and opposition-specific

strengths, ensuring that the machine learning models can effectively differentiate player capabilities.

3.1.2 Performance Scoring System

Performance scores are computed using weight values derived from machine learning models, ensuring an objective and data-driven approach to player evaluation. The system assesses player performance across different leagues and assigns a weighted score based on key parameters such as runs, averages, wickets, and consistency. By leveraging machine learning techniques, we optimize the scoring mechanism to reflect a player's true potential.

The process begins with data preparation, where data is scraped, transformed, and cleaned from various sources to ensure accuracy and consistency. Following this, feature selection is conducted to extract key performance indicators (KPIs) for each player, focusing on metrics that best represent their capabilities. Machine learning models are then applied to determine feature importance, analyzing which factors contribute most to player performance[1].

To ensure the most reliable scoring system, model evaluation is performed by testing multiple models and selecting the best one based on Mean Squared Error (MSE) and R^2 Score. The final step involves score calculation, where the feature importance values from the best-performing model are used to derive a weighted equation for performance scoring. This method ensures that player selection is based on empirical data rather than subjective judgment, leading to more accurate and strategic team composition.

IBaC	0.5*Inns + 0.44*Ave + 0.05*50s + 0.01*SR + 0.04*100s
IBoC	0.77*Overs + 0.01*Inns + 0.16*SR + 0.04*Ave + 0.03*4or5 wickts
LBaC	0.47*Inns + 0.30*Ave + 0.41*50s + 0.047*SR + 0.017*100s
LBoC	0.70*Overs + 0.11*Runs + 0.06*Ave + 0.03*Inns + 0.01*4or5 wickts
IAC	IBaC + IBoC
LAC	LBaC + LBoC

Table ii: Performance Score Equations

We computed the International Score (Intl Score), LPL Score, Venue Score, and Fan Score using the following approach:

- Intl Score:
    - If the player is a batter → IBaC (International Batting Contribution) was used
    - If the player is a bowler → IBoC (International Bowling Contribution) was used
    - If the player is an all-rounder → IAC (International All-rounder Contribution) was used
- LPL Score:
    - If the player is a batter → LBaC (LPL Batting Contribution) was used
    - If the player is a bowler → LBoC (LPL Bowling Contribution) was used
    - If the player is an all-rounder → LAC (LPL All-rounder Contribution) was used
- Venue Score:
    - We extracted venue-related IBaC, IBoC, and IAC values to assess player adaptability at different venues.
- Fan Score:
    - Calculated based on the frequency of selections by fans in our dataset, representing the player's popularity and perceived value in the squad.

Using these values, we developed a Fitness Score, which varies based on playing conditions and locations. The weight values were determined using the same methodology applied during performance score generation. Below is the table summarizing the fitness score calculations:

Team	Fitness Score Calculation
Main	0.5 x [Intl Score] + 0.4 x [LPL Score] + 0.1 x[Fan Score]
New Zealand	0.36*Intl Score + 0.02*LPL Score + 0.43*Venue Score + 0.17*Fan Score
Australia	0.34*Intl Score + 0.12*LPL Score + 0.41*Venue Score + 0.12*Fan Score
Sri Lanka	0.28*Intl Score + 0.25*LPL Score + 0.31*Venue Score + 0.14*Fan Score
England	0.35*Intl Score + 0.08*LPL Score + 0.39*Venue Score + 0.17*Fan Score
India	0.32*Intl Score + 0.22*LPL Score + 0.23*Venue Score + 0.21*Fan Score
Bangladesh	0.31*Intl Score + 0.21*LPL Score + 0.27*Venue Score + 0.19*Fan Score
West Indies	0.29*Intl Score + 0.18*LPL Score + 0.33*Venue Score + 0.18*Fan Score
South Africa	0.36*Intl Score + 0.02*LPL Score + 0.43*Venue Score + 0.17*Fan Score

Table 3: Fitness Score Calculation

3.1.3 Squad Optimization using Genetic Algorithm

A Genetic Algorithm (GA) is implemented to optimize squad selection based on computed performance scores. The algorithm applies selection, crossover, and mutation techniques to evolve an optimal playing XI.[3].

The squad optimization process was implemented using a Genetic Algorithm (GA) to iteratively refine the selection of the best playing XI. The process began with an initial population of randomly generated teams. Each team was evaluated based on fitness criteria, which included performance scores and team role balance to ensure a well-structured squad composition. The fitness evaluation determined whether the current population contained an optimal solution. If the optimal squad was not achieved, selection, crossover, and mutation operations were applied to generate a new population, enhancing diversity and improving team composition with each iteration. The selection step retained the fittest individuals, crossover combined traits from different squads, and mutation introduced small variations to explore potential improvements. This cycle continued until an optimal solution was found, at which point the final team was determined. The final output included both a main playing XI and

venue-specific teams, ensuring adaptability based on match conditions. This GA-driven approach enabled a data-driven and dynamic selection of the best squad.

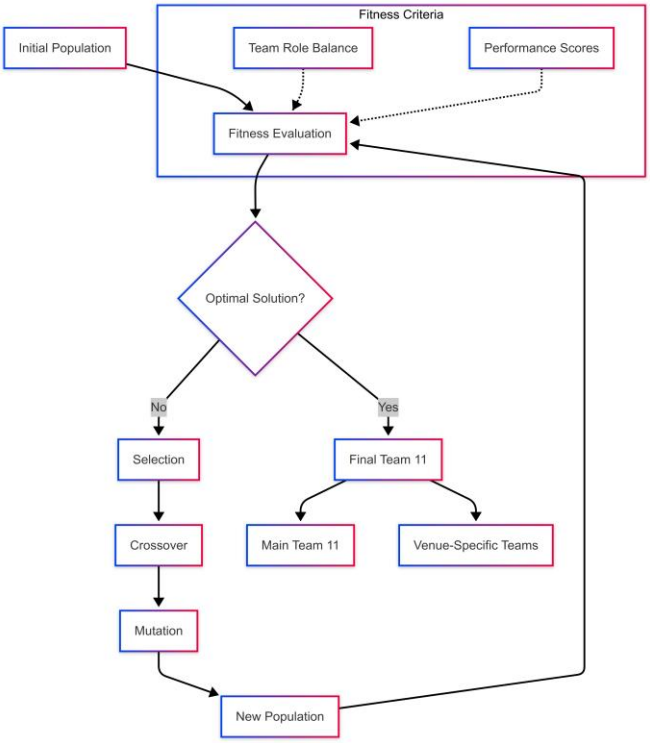


Figure ii: Genetic Algorithm Flow

3.1.4 Venue Prediction Model

The venue prediction model was developed by analyzing past match data, incorporating the playing XI from previous and created a feature labeled 'venue,' while other features represented player names[1]. The venue prediction model was designed through a structured machine learning pipeline, beginning with data collection from historical match records, which included previous team selections and venue details. In the data preparation phase, multiple datasets were combined, and label encoding was applied to player names and venues to convert categorical data into a numerical format. Feature engineering techniques were used to enhance the dataset before splitting it into training and testing sets. Various machine learning models, including K-Nearest Neighbors (KNN), Random Forest, Support Vector Machine, Logistic Regression, and Decision Tree, were trained and evaluated based on key performance metrics such as accuracy, precision, recall, and F1-score. Following a thorough evaluation, the KNN model demonstrated the best performance and was selected as the final model for deployment. The trained model was then integrated into the web application to provide venue-based squad selection predictions. You can see the corresponding data flow diagram below[4].

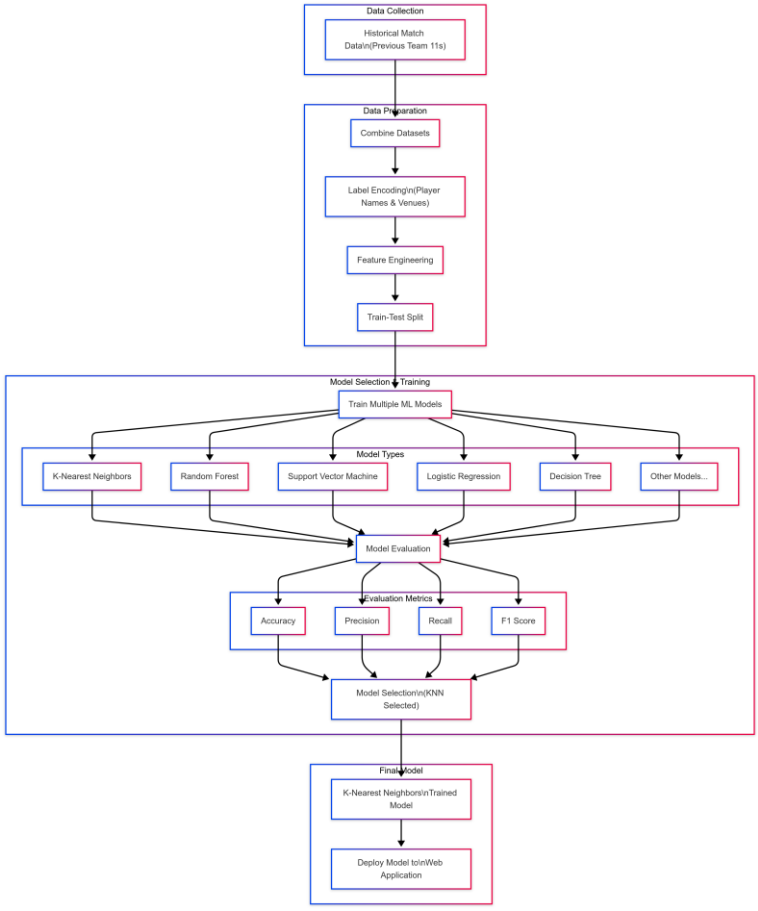


Figure iii: Venue Prediction Model Data flow

3.1.5 Fan-Based Selection Probability Model

The Fan-Based Selection Probability Model was developed to analyze fan preferences for selecting the best playing XI. Data was collected through Google Forms, which were distributed via social media platforms like Facebook and Twitter, as well as WhatsApp groups. The collected responses contained preferred player selections from fans. During data processing, the raw selection data was cleaned, and a favoritism score was calculated based on selection frequency. Label encoding was applied to player names, followed by feature engineering and dataset splitting into training and testing sets. Multiple machine learning models, including XGBoost, Random Forest, Linear Regression, and Decision Tree, were trained and evaluated using metrics such as Mean Absolute Error, Mean Squared Error, Root Mean Squared Error, R^2 Score, and Accuracy. After thorough evaluation, XGBoost was selected as the final model due to its superior performance. The trained model was then deployed in the web application under the fan tab, enabling probability-based player selection predictions. The model flow diagram is shown below.

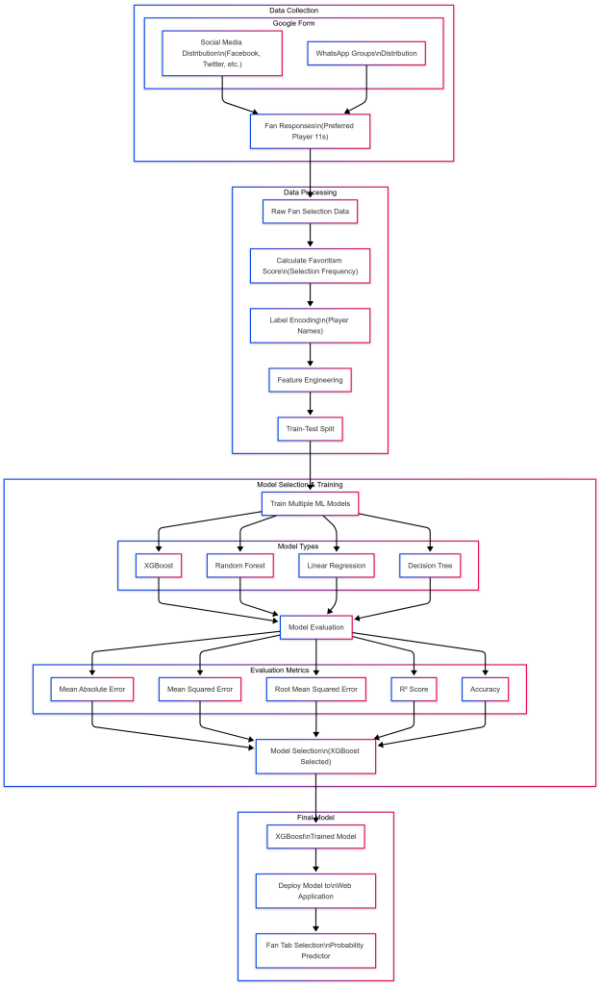


Figure iv: Fan Based Selection Probability Model Data flow

3.1.6 Web Application & AI Chatbot

The final system is deployed as a web application, enabling users to seamlessly view the selected squad, explore team compositions based on different selection criteria, and interact with an AI-powered chatbot for cricket-related insights. This platform was designed with a modular and interconnected architecture, ensuring efficient data processing and seamless user interaction across various components.

At its core, the system collects and processes data through advanced machine learning models, which then power multiple frontend interfaces, including the Main Dashboard, International Tab, and LPL Tab. The backend architecture was developed to support a range of critical functionalities, such as the Team Selection Algorithm and the RAG Chatbot System. This structured approach facilitates smooth navigation between key features like the

Optimal Team 11 Display, Venue Selection Tool, International Performance Metrics, Player Rankings, and LPL Performance Metrics.

Beyond these core capabilities, additional specialized features were integrated to enhance the platform's analytical depth. These include Domestic Cricket Analysis, a Fan-Selected Team 11 module, and a Selection Probability Calculator, all of which contribute to a data-driven approach to squad selection. The chatbot interface was designed to handle both direct query inputs and cricket-related information display, further expanding the application's utility. Additionally, School Cricket Tournament Data was incorporated, enriching the system with grassroots-level cricket insights.

This holistic design ensures a dynamic and engaging user experience, where data flows seamlessly across different sections of the application. By combining advanced analytics with intuitive user interactions, the platform delivers comprehensive cricket insights tailored to various selection criteria and performance evaluations.

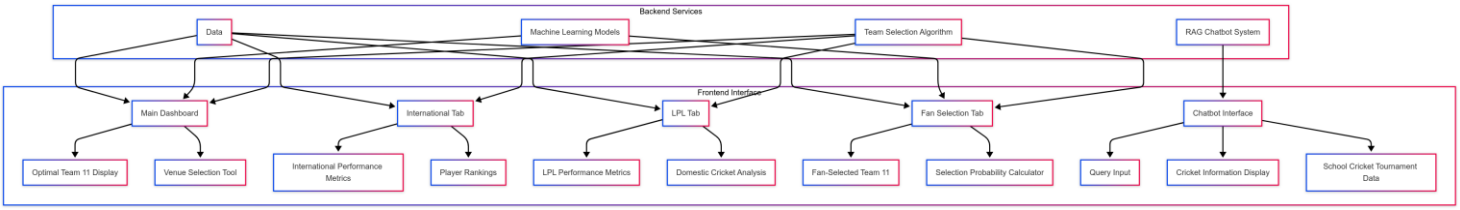


Figure v: Web Application

3.2 Methodological Approach

We have adopted a data-driven and machine learning-based approach to enhance squad selection, ensuring an objective and optimized team composition. The methodology follows a structured sequence of steps, beginning with data collection, where player statistics are scraped and aggregated from ESPN, Papare, and Google Forms. This ensures a diverse dataset that includes international performance metrics, domestic league statistics, and fan-based selection preferences.

Once the data is gathered, feature engineering is applied to extract and transform key performance indicators (KPIs) that define player capabilities. This step refines raw data into meaningful attributes that influence selection decisions. Following this, performance score calculation is conducted using machine learning models to compute weighted scores. These scores are derived by analyzing player performance across various metrics such as runs, averages, wickets, and consistency.

To optimize squad selection, a genetic algorithm is implemented, ensuring that the best possible team is selected based on computed performance scores. Additionally, model training and evaluation are performed to develop predictive models for venue-based selection and fan voting impact, enabling dynamic team adjustments based on playing conditions and public preferences.

Finally, all these components are integrated into an interactive web application, allowing users to explore and analyze different squad selection scenarios. This platform provides real-time insights, making the selection process more transparent and data-driven.

3.3 Limitations, Assumptions, and Considerations

The development of the cricket analytics web application encountered several critical limitations that necessitate careful consideration. The accuracy of squad selection fundamentally hinges on the quality and completeness of available data, recognizing that machine learning models inherently rely on historical performance metrics which may not adequately capture sudden shifts in player form or unexpected performance variations. The fan voting model introduces an additional layer of complexity, predicated on the assumption of a truly representative sample of cricket enthusiasts, which may not always reflect the nuanced realities of player capabilities.

Underlying the system's design were key assumptions about performance metrics serving as an accurate proxy for player ability, and a belief that fan votes could meaningfully influence selection decisions through a structured mechanism. However, these assumptions are not without potential pitfalls. The research team anticipated foreseeable challenges, particularly the potential unavailability of consistent data from all sources. To mitigate this risk, synthetic data generation techniques were proposed as an innovative alternative, capable of filling gaps in datasets and ensuring more robust analytical models.

Another significant challenge emerged around potential biases inherent in fan voting patterns. Recognizing that fan sentiment can be subjective and potentially skewed, the team developed a sophisticated alternative approach: implementing a weighted voting mechanism designed to balance and neutralize these inherent biases. This approach would allow for more nuanced and balanced player selection, ensuring that fan input remains valuable while preventing any single demographic or passionate fan base from disproportionately influencing team composition decisions.

4. Experimental Setup and Implementation

4.1 User Interface

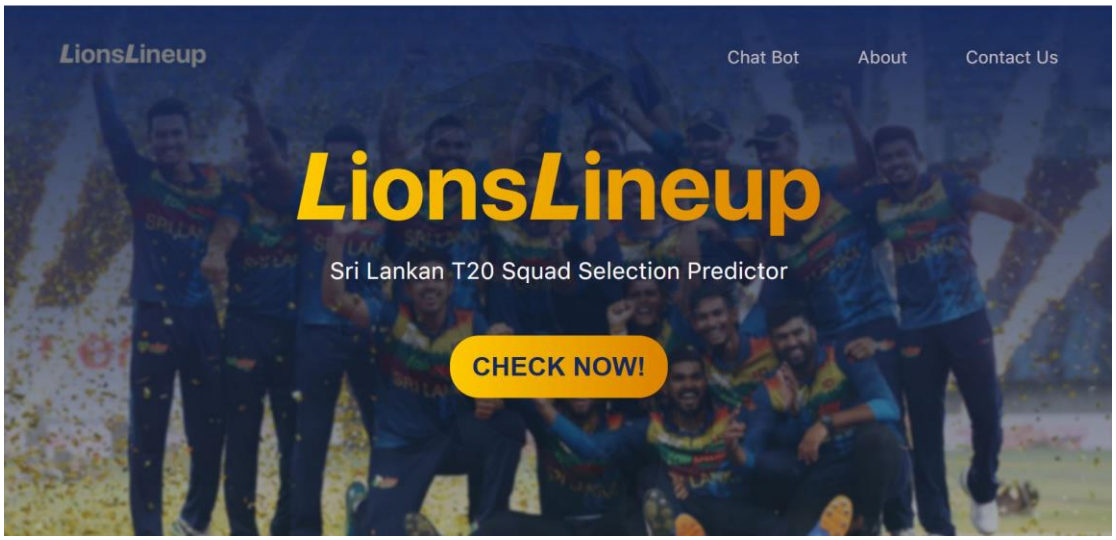


Figure vi: UI Design

We designed the web application to be intuitive and feature-rich, displaying selected squads, allowing users to interact with different selection criteria, and offering AI-driven cricket insights.

4.2 Prototype Functionality

The prototype integrates multiple functionalities to enhance squad selection and user interaction. The system is designed to aggregate, clean, and process data from various cricket sources, ensuring a comprehensive evaluation of player performance. A weighted performance score is computed using machine learning-derived feature importance values, facilitating an objective assessment of player capabilities. To optimize squad selection, a genetic algorithm is employed to identify the best possible playing XI, ensuring that team composition is balanced and strategically sound. Additionally, the system incorporates venue-based selection by leveraging historical match data and machine learning models to predict the most suitable squad for a given venue.

A fan-based prediction component estimates the probability of a player's selection by analyzing fan voting data. This aspect enhances the interactive nature of the system, allowing public sentiment to be reflected in squad formation. The web interface is designed to be user-friendly, providing intuitive navigation and real-time AI-driven cricket insights. To further assist users, an AI chatbot is integrated to answer cricket-related queries and provide squad information, enhancing the overall user experience. The prototype successfully demonstrates the Smart Squad Selection System's ability to integrate advanced data-driven methodologies for improved decision-making in Sri Lankan cricket team selection.

4.3 Data Manipulation and Testing

4.3.1 Data Collection Process

To build a robust selection model, data was collected from multiple sources to ensure a comprehensive assessment of player performance. The first step involved web scraping, where match statistics were extracted from platforms such as ESPN, Papare, and Cricbuzz using Python-based techniques. This allowed for the collection of real-time and historical data, providing a detailed understanding of player contributions in various formats.

In addition to match statistics, fan-based data collection was conducted to incorporate public opinion into the selection process. This was achieved through Google Forms, where fans submitted their votes for preferred players. By aggregating these responses, the model was able to gauge popular sentiment and integrate it into the overall player evaluation.

Furthermore, detailed match performance data was recorded to capture key performance indicators that reflect a player's effectiveness on the field. Metrics such as batting averages, strike rates, wickets taken, and economy rates were carefully analyzed to quantify individual contributions. By integrating these diverse data sources, the model ensured a holistic approach to squad selection, balancing statistical performance with fan preferences.

4.3.2 Preprocessing Steps

To ensure high data quality, several preprocessing steps were applied before training the machine learning models. The first step was data cleaning, where missing values were handled through imputation techniques, duplicate records were removed, and inconsistencies in player

names, match formats, and statistics were resolved. This step ensured that the dataset remained accurate and reliable for further analysis.

Next, data formatting was carried out to standardize numerical values and structure the dataset appropriately for machine learning model training. Player statistics, such as strike rates and economy rates, were normalized to maintain consistency across different formats and leagues. Proper structuring of categorical data, such as player roles and match conditions, was also performed to enhance the efficiency of model processing.

Additionally, feature engineering played a crucial role in refining the dataset for better predictive accuracy. Key attributes such as venue-based performance, recent form, and opposition-specific strengths were extracted to provide deeper insights into a player's adaptability. These engineered features helped improve the model's ability to evaluate player potential in different playing conditions, leading to more accurate squad selection.

4.3.3 Performance Score Calculation

Performance scores were computed based on six key metrics:

- IBaC
- IBoC
- IAC
- LBaC
- LBoC
- LAC

```
[ ] bowling_weights = {
        'Overs': 0.772074,
        'Inns': 0.013616,
        'SR': 0.164369,
        'Ave': 0.037204,
        '4or5 wickts': 0.033091
    }
```

```
[ ] data['BPS'] = (
        (bowling_weights['Overs'] * data['Overs']) +
        (bowling_weights['Inns'] * data['Inns']) +
        (bowling_weights['SR'] * data['SR']) +
        (bowling_weights['Ave'] * data['Ave']) +
        (bowling_weights['4or5 wickts'] * data['4or5 wickts'])
    )
```

Figure vii: Performance Score Calculation Code

To derive an accurate and reliable performance score equation, machine learning models were employed to analyze player data and generate predictive insights. Various regression-based techniques were explored, ensuring that the final model provided the best balance between predictive accuracy and interpretability. The best-performing model was selected based on Mean Squared Error (MSE) and R^2 Score, optimizing both precision and explanatory power.

To identify the most influential factors affecting performance, feature importance analysis was conducted using the following methods:

Random Forest Regression: A robust ensemble learning method that evaluates feature importance based on the frequency of splits across multiple decision trees, providing insights into which attributes contribute most to performance prediction.

XGBoost Regression: An advanced gradient-boosting technique that enhances prediction accuracy by iteratively improving upon previous errors while ranking feature significance.

Lasso Regression: A linear regression model with L1 regularization that helps in feature selection by reducing the coefficients of less important variables to zero, effectively filtering out irrelevant attributes.

Permutation Importance: A model-agnostic approach that measures the impact of each feature by shuffling its values and observing the subsequent change in model performance, ensuring unbiased feature ranking.

By leveraging these methodologies, the most critical factors influencing player performance were identified, allowing for the development of a data-driven selection model that optimally balances different performance attributes.

```
[ ]  feature_importances_df = pd.DataFrame({
         "Random Forest": rf_importances,
         "XGBoost": xgb_importances,
         "Lasso Regression": lasso_importances,
         "Permutation Importance": permutation_importances
     })
```

```
[ ]  print(
         feature_importances_df.head(10)
     )
```

Figure viii: Weight Value Generation Using ML code part

4.3.4 Machine Learning Model Training

Venue Prediction Model

To build an effective selection model, historical match data was analyzed, focusing on 40 winning team combinations to understand the patterns of successful team compositions. This dataset provided valuable insights into player contributions and strategic factors influencing match outcomes.

For machine learning model training, player names and venue details were encoded into numerical representations. This transformation was necessary to ensure the data was in a structured format suitable for training predictive models. By encoding categorical variables, the model could recognize patterns and relationships between different features, enhancing its ability to make accurate predictions.

To identify the most suitable predictive approach, multiple machine learning models were evaluated, including Random Forest, Decision Tree, Logistic Regression, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN). Each model was assessed based on its ability to classify successful team combinations effectively.

After rigorous evaluation, K-Nearest Neighbors (KNN) emerged as the optimal model due to its superior performance across key metrics such as accuracy, precision, recall, and F1-score. The KNN model demonstrated high predictive reliability in determining the best team compositions based on historical data, making it the most suitable choice for the final selection framework.

```python
# Simplify labels: Group teams by fitness score ranges (e.g., 0-30, 31-60, 61-100)
df['Team_Group'] = pd.cut(df['Fitness_Score'], bins=[0, 2, 3, 5], labels=['Low', 'Medium', 'High'])

# Update target variable
y = df['Team_Group']

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train models again
results_grouped = {}
for model_name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    results_grouped[model_name] = {
        "Accuracy": accuracy,
        "Classification Report": classification_report(y_test, y_pred, zero_division=0),
        "Confusion Matrix": confusion_matrix(y_test, y_pred)
    }

# Display grouped results
for model_name, result in results_grouped.items():
    print(f"Model: {model_name}")
    print(f"Accuracy: {result['Accuracy']:.4f}")
    print(result["Classification Report"])
    print(result["Confusion Matrix"])
    print("-" * 50)
```

Figure ix: Venue prediction model code

Fan-Based Selection Model

To incorporate public opinion into the selection model, fan vote data was collected and systematically processed into a favoritism score. This score quantified the popularity of each player based on the frequency of selections by fans, providing an additional dimension to player evaluation beyond statistical performance.

For machine learning implementation, player names were label-encoded, converting categorical values into numerical representations. This transformation ensured compatibility with machine learning models, enabling them to analyze and predict player selections effectively. Several models, including Random Forest, Linear Regression, XGBoost, and Decision Tree, were tested to determine the most accurate prediction framework.

After evaluating performance using key metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), R^2 Score, and accuracy, XGBoost was selected as the best-performing model. XGBoost's robust feature selection and handling of non-linearity contributed to its superior predictive capabilities, making it the optimal choice for modeling favoritism scores in the final selection system.

```python
models = {
    'Random Forest': RandomForestRegressor(n_estimators=100, random_state=42),
    'Linear Regression': LinearRegression(),
    'XGBoost': XGBRegressor(n_estimators=100, random_state=42),
    'Decision Tree': DecisionTreeRegressor(random_state=42)
}


threshold = 0.1


results = {}

for model_name, model in models.items():
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)


    mae = mean_absolute_error(y_test, predictions)
    mse = mean_squared_error(y_test, predictions)
    rmse = mse ** 0.5
    r2 = r2_score(y_test, predictions)


    relative_error = np.abs((predictions - y_test) / y_test)
    accuracy = np.mean(relative_error < threshold) * 100  # Percentage accuracy


    results[model_name] = {
        'MAE': mae,
        'MSE': mse,
        'RMSE': rmse,
        'R² Score': r2,
        'Accuracy': accuracy
    }


for model_name, metrics in results.items():
    print(f"\nModel: {model_name}")
    for metric_name, metric_value in metrics.items():
        print(f"{metric_name}: {metric_value:.4f}")
```

Figure x: Fan based selection model code

4.3.5 Genetic Algorithm Implementation

The squad optimization module was developed using a Genetic Algorithm (GA) to identify the best possible playing XI. This approach leveraged evolutionary principles to iteratively refine squad selection, ensuring optimal team composition based on player performance data.

The process began with Population Initialization, where a randomly generated set of squads was created as the initial pool of candidates. Each squad was then assessed through Fitness Evaluation, which involved computing an overall performance score derived from individual player statistics. Based on these scores, the Selection phase identified high-performing squads for further refinement, ensuring that only the most promising candidates progressed.

To enhance diversity and improve squad configurations, the Crossover step combined players from selected squads to generate new variations. Additionally, Mutation was introduced by making small random modifications to squads, preventing premature convergence and

maintaining a diverse set of potential solutions. This iterative process continued until an optimized squad was achieved, ensuring that the final selection represented the best possible combination of players based on predefined performance metrics.

```python
def genetic_algorithm(population_size=50, generations=100, mutation_rate=0.1, crossover_rate=0.8):
    population = [create_individual() for _ in range(population_size)]

    for generation in range(generations):
        fitness_scores = [calculate_fitness(individual) for individual in population]
        population = [x for _, x in sorted(zip(fitness_scores, population), reverse=True)]

        new_population = population[:population_size // 2]

        while len(new_population) < population_size:
            parent1, parent2 = random.sample(population[:population_size // 2], 2)

            if random.random() < crossover_rate:
                child1, child2 = crossover(parent1, parent2)
            else:
                child1, child2 = parent1, parent2

            if random.random() < mutation_rate:
                child1 = mutate(child1)
            if random.random() < mutation_rate:
                child2 = mutate(child2)

            new_population.extend([child1, child2])

        population = new_population[:population_size]

        # Validate that all players in each team are unique
        for individual in population:
            player_ids = [player.PlayerID for player in individual]
            assert len(player_ids) == len(set(player_ids)), f"Error: Duplicate players found in team {individual}"

        print(f"Generation {generation + 1}: Best Fitness = {max(fitness_scores)}")

    return population[0]
```

Figure xi: Genetic algorithm code

4.3.6 RAG AI Chatbot

To enhance user interaction and provide real-time cricket insights, an AI-powered chatbot was integrated into the system. This chatbot leverages natural language processing (NLP) to answer user queries and retrieve cricket-related insights efficiently. The chatbot's functionality is powered by a retrieval-augmented generation (RAG) approach, ensuring accurate and context-aware responses by combining retrieval-based knowledge with generative AI capabilities.

The chatbot was built using FastAPI, a high-performance web framework that facilitates the creation of API endpoints for seamless communication between the user interface and backend

processing. LangChain was employed for document retrieval and processing, enabling the chatbot to extract relevant information from cricket-related datasets. For efficient and scalable vector-based retrieval, FAISS (Facebook AI Similarity Search) was implemented, allowing rapid searching and ranking of relevant cricket documents based on user queries. To enhance query representation and improve the chatbot's understanding of cricket-related contexts, Google Generative AI Embeddings were utilized, ensuring that queries are mapped effectively to relevant knowledge sources.

This AI-powered chatbot serves as a valuable tool for users, offering real-time cricket insights such as squad selection recommendations, historical player performances, and venue-based statistics. By integrating advanced machine learning techniques and NLP capabilities, the chatbot significantly enhances user experience, providing data-driven and context-aware responses to cricket enthusiasts.

```python
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel
import os
from contextlib import asynccontextmanager
from fastapi.middleware.cors import CORSMiddleware
from langchain_groq import ChatGroq
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.chains.combine_documents import create_stuff_documents_chain
from langchain_core.prompts import ChatPromptTemplate
from langchain.chains import create_retrieval_chain
from langchain_community.vectorstores import FAISS
from langchain_community.document_loaders import PyPDFDirectoryLoader
from langchain_google_genai import GoogleGenerativeAIEmbeddings
from dotenv import load_dotenv
import time

load_dotenv()

# Global variable for vector store
vectors = None


@asynccontextmanager
async def lifespan(app: FastAPI):
    # Load documents on startup
    global vectors
    embeddings = GoogleGenerativeAIEmbeddings(model="models/embedding-001")
    loader = PyPDFDirectoryLoader("./us_census")
    docs = loader.load()
    text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=200)
    final_documents = text_splitter.split_documents(docs[:20])
    vectors = FAISS.from_documents(final_documents, embeddings)
```

Figure xii: RAG AI Chatbot code

4.4 Pitfalls and Workarounds

During the research and development process, several challenges were encountered, necessitating innovative solutions. Data collection issues arose due to inconsistencies in web

scraping. These were mitigated by implementing error-handling mechanisms and automated retry functions, ensuring continuous data flow. Feature selection proved complex, as different machine learning models produced varied results. To address this, multiple models were tested to determine the optimal feature selection and score computation approach.

Computational limitations were another significant hurdle, particularly in the genetic algorithm's optimization process. To improve efficiency, optimization techniques were applied, reducing processing time while maintaining accuracy. The chatbot's response accuracy also required fine-tuning, as initial retrieval mechanisms were not always precise. FAISS-based vector retrieval was refined to enhance relevance, resulting in more accurate and contextually appropriate responses.

4.5 Summary

This chapter detailed the experimental setup and implementation of the research. Key aspects, including data collection, preprocessing, machine learning modeling, genetic algorithm optimization, and AI chatbot integration, were covered. The next chapter will discuss the results and analysis of the implemented system.

5 Results and Analysis

5.1 Team Player Role Distribution

This section presents the optimal team composition based on the number of batters, bowlers, and all-rounders within the selected squad. The distribution is analyzed both for the main optimal XI and for venue-specific teams to ensure balance and effectiveness.

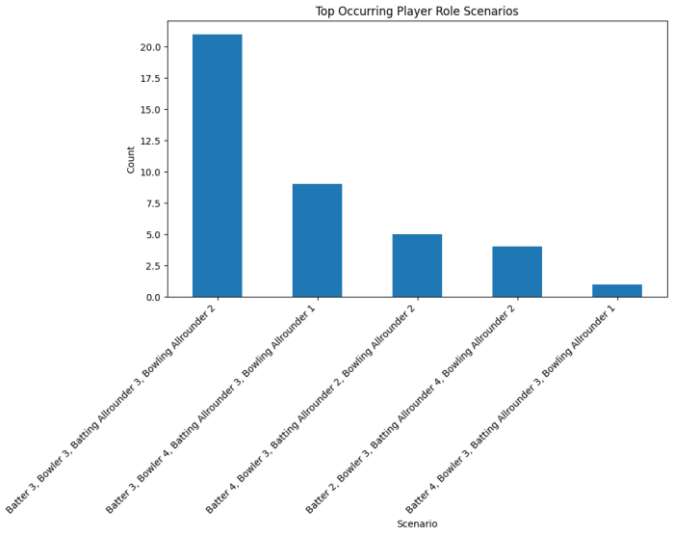


Figure xiv: Player Distribution according to Role- Main

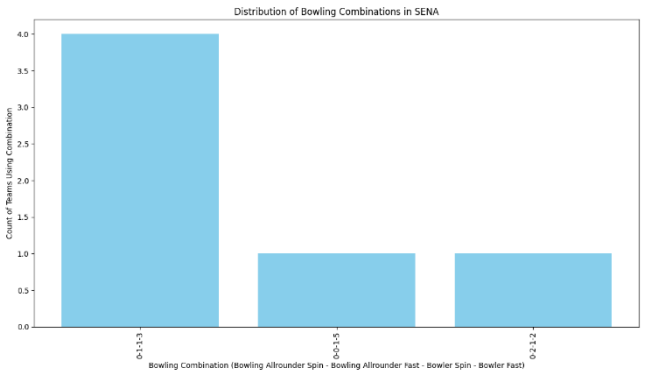


Figure xiii: Player Distribution according to Role- Sena

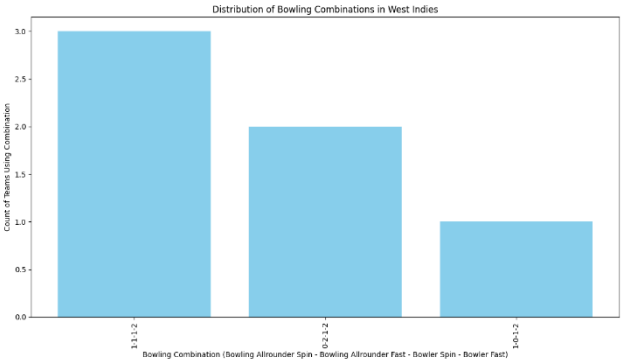


Figure xvi: Player Distribution according to Role- WestIndies

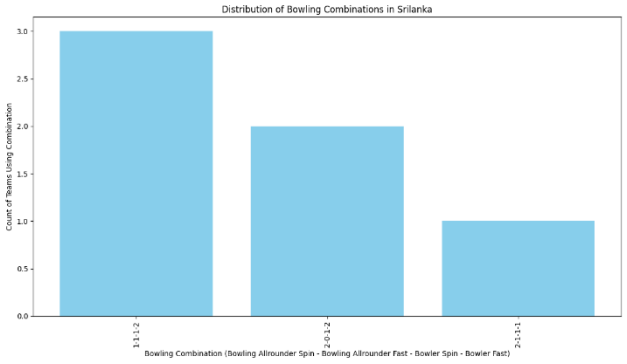


Figure xv: Player Distribution according to Role- Sri Lanka

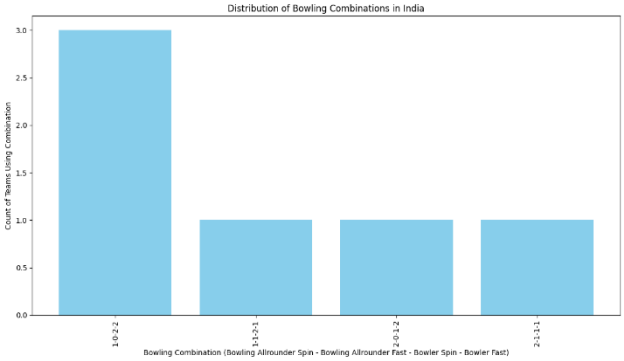


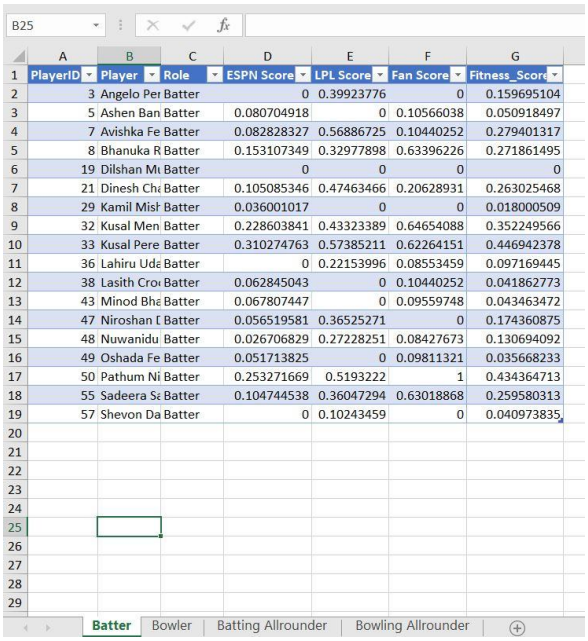
Figure xvii: Player Distribution according to Role- India

Sheet	Bowling Allrounder Spin	Bowling Allrounder Fast	Bowler Spin	Bowler Fast
SENA	0	1	1	3
Srilanka	1	1	1	2
West indies	1	1	1	2
India	1	0	2	2

Figure xviii: Bowler & Bowling Allrounder Distribution according to Role

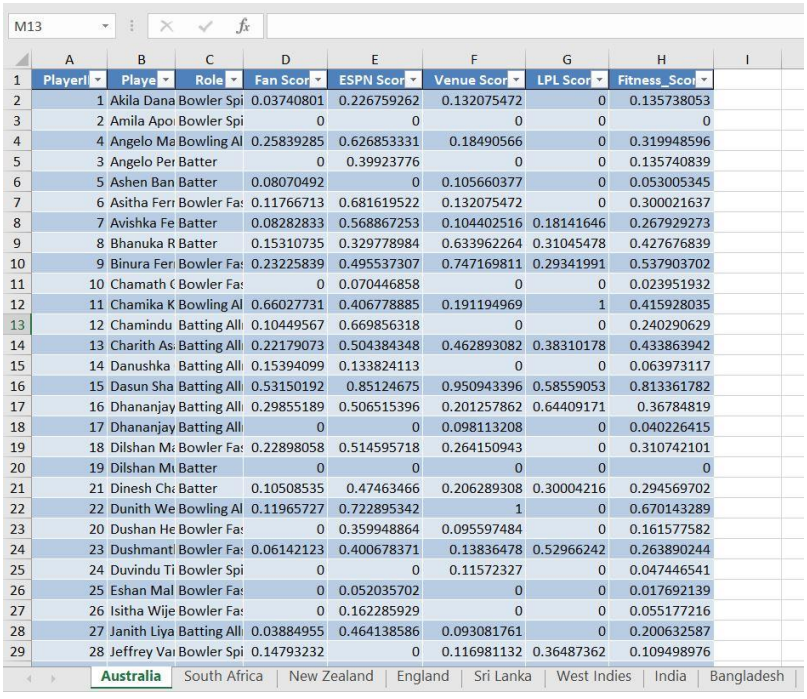
5.2 Fitness Score Calculation

Using these performance scores, we computed the International Score (Intl Score), Venue Score, and Fan Score for each player. Based on these values, a Fitness Score was created using weighted formulas tailored for different conditions and locations. The weight values were determined using the same methodology applied during performance score generation. Below is the table summarizing the fitness scores for different playing conditions.



PlayerID	Player	Role	ESPN Score	LPL Score	Fan Score	Fitness_Score
3	Angelo Per	Batter	0	0.39923776	0	0.159695104
5	Ashen Ban	Batter	0.080704918	0	0.10566038	0.050918497
7	Avishka Fe	Batter	0.082828327	0.56886725	0.10440252	0.279401317
8	Bhanuka R	Batter	0.153107349	0.32977898	0.63396226	0.271861495
19	Dilshan Mı	Batter	0	0	0	0
21	Dinesh Cha	Batter	0.105085346	0.47463466	0.20628931	0.263025468
29	Kamil Mish	Batter	0.036001017	0	0	0.018000509
32	Kusal Men	Batter	0.228603841	0.43323389	0.64654088	0.352249566
33	Kusal Pere	Batter	0.310274763	0.57385211	0.62264151	0.446942378
36	Lahiru Uda	Batter	0	0.22153996	0.08553459	0.097169445
38	Lasith Cro	Batter	0.062845043	0	0.10440252	0.041862773
43	Minod Bha	Batter	0.067807447	0	0.09559748	0.043463472
47	Niroshan D	Batter	0.056519581	0.36525271	0	0.174360875
48	Nuwanidu	Batter	0.026706829	0.27228251	0.08427673	0.130694092
49	Oshada Fe	Batter	0.051713825	0	0.09811321	0.035668233
50	Pathum Ni	Batter	0.253271669	0.5193222	1	0.434364713
55	Sadeera Sa	Batter	0.104744538	0.36047294	0.63018868	0.259580313
57	Shevon Da	Batter	0	0.10243459	0	0.040973835

Figure xix: Fitness Scores Main



PlayerI	Playe	Role	Fan Scor	ESPN Scor	Venue Scor	LPL Scor	Fitness_Scor
1	Akila Dana	Bowler Spi	0.03740801	0.226759262	0.132075472	0	0.135738053
2	Amila Apo	Bowler Spi	0	0	0	0	0
4	Angelo Ma	Bowling Al	0.25839285	0.626853331	0.18490566	0	0.319948596
3	Angelo Per	Batter	0	0.39923776	0	0	0.135740839
5	Ashen Ban	Batter	0.08070492	0	0.105660377	0	0.053005345
6	Asitha Fern	Bowler Fas	0.11766713	0.681619522	0.132075472	0	0.300021637
7	Avishka Fe	Batter	0.08282833	0.568867253	0.104402516	0.18141646	0.267929273
8	Bhanuka R	Batter	0.15310735	0.329778984	0.633962264	0.31045478	0.427676839
9	Binura Fer	Bowler Fas	0.23225839	0.495537307	0.747169811	0.29341991	0.537903702
10	Chamath C	Bowler Fas	0	0.070446858	0	0	0.023951932
11	Chamika K	Bowling Al	0.66027731	0.406778885	0.191194969	1	0.415928035
12	Chamindu	Batting All	0.10449567	0.669856318	0	0	0.240290629
13	Charith As	Batting All	0.22179073	0.504384348	0.462893082	0.38310178	0.433863942
14	Danushka	Batting All	0.15394099	0.133824113	0	0	0.063973117
15	Dasun Sha	Batting All	0.53150192	0.85124675	0.950943396	0.58559053	0.813361782
16	Dhananjay	Batting All	0.29855189	0.506515396	0.201257862	0.64409171	0.36784819
17	Dhananjay	Batting All	0	0	0.098113208	0	0.040226415
18	Dilshan Ma	Bowler Fas	0.22898058	0.514595718	0.264150943	0	0.310742101
19	Dilshan Mı	Batter	0	0	0	0	0
21	Dinesh Cha	Batter	0.10508535	0.47463466	0.206289308	0.30004216	0.294569702
22	Dunith We	Bowling Al	0.11965727	0.722895342	1	0	0.670143289
20	Dushan He	Bowler Fas	0	0.359948864	0.095597484	0	0.161577582
23	Dushmant	Bowler Fas	0.06142123	0.400678371	0.13836478	0.52966242	0.263890244
24	Duvindu Ti	Bowler Spi	0	0	0.11572327	0	0.047446541
25	Eshan Mal	Bowler Fas	0	0.052035702	0	0	0.017692139
26	Isitha Wije	Bowler Fas	0	0.162285929	0	0	0.055177216
27	Janith Liya	Batting All	0.03884955	0.464138586	0.093081761	0	0.200632587
28	Jeffrey Van	Bowler Spi	0.14793232	0	0.116981132	0.36487362	0.109498976

Figure xx: Venue specific Fitness Scores

5.3 Genetic Algorithm

A Genetic Algorithm was implemented to generate the best squad selection for both the main optimal XI and venue-specific optimal XIs. The algorithm iteratively selects, mutates, and evolves potential team combinations to find the best selection based on performance scoring[3].

```
Best Team Selected:
Pathum Nissanka (Batter) - Fitness: 0.4343647126694775
Kusal Perera (Batter) - Fitness: 0.4469423782372291
Avishka Fernando (Batter) - Fitness: 0.27940131670286794
Maheesh Theekshana (Bowler) - Fitness: 0.8268221573661552
Dilshan Madushanka (Bowler) - Fitness: 0.3967436715058713
Binura Fernando (Bowler) - Fitness: 0.3590610968673313
Dasun Shanaka (Batting Allrounder) - Fitness: 0.6613440022468193
Kamindu Mendis (Batting Allrounder) - Fitness: 0.4904141804810641
Angelo Mathews (Batting Allrounder) - Fitness: 0.3984283254035007
Chamika Karunaratne (Bowling Allrounder) - Fitness: 0.4939697066897755
Wanindu Hasaranga (Bowling Allrounder) - Fitness: 0.9928930817610062
```

Figure xxix: GA Best Team Selection- Main

```
Pathum Nissanka (Batter) - Fitness: 0.5225063732040306
Kusal Mendis (Batter) - Fitness: 0.5009197839477646
Kusal Perera (Batter) - Fitness: 0.4876257092385576
Charith Asalanka (Batting Allrounder) - Fitness: 0.4338639423197982
Dasun Shanaka (Batting Allrounder) - Fitness: 0.8133617816431096
Kamindu Mendis (Batting Allrounder) - Fitness: 0.6768346538795146
Chamika Karunaratne (Bowling Allrounder Fast) - Fitness: 0.4159280353811305
Maheesh Theekshana (Bowler Spin) - Fitness: 0.7874977194978996
Nuwan Thushara (Bowler Fast) - Fitness: 0.5163585619856205
Binura Fernando (Bowler Fast) - Fitness: 0.5379037019937127
```

Figure xxviii: GA Best Team Selection- Aus

```
Kusal Perera (Batter) - Fitness: 0.4369728863032465
Kusal Mendis (Batter) - Fitness: 0.3752805543018121
Pathum Nissanka (Batter) - Fitness: 0.4169202453147067
Kamindu Mendis (Batting Allrounder) - Fitness: 0.5660968574334364
Dasun Shanaka (Batting Allrounder) - Fitness: 0.6865366426757022
Charith Asalanka (Batting Allrounder) - Fitness: 0.3836936893705808
Wanindu Hasaranga (Bowling Allrounder Spin) - Fitness: 0.9749811320754718
Dunith Wellalage (Bowling Allrounder Spin) - Fitness: 0.5151247049659343
Maheesh Theekshana (Bowler Spin) - Fitness: 0.7602489796924542
Binura Fernando (Bowler Fast) - Fitness: 0.3917156705481141
Nuwan Thushara (Bowler Fast) - Fitness: 0.39933746129736
```

Figure xxvii: GA Best Team Selection- Ban

```
Kusal Mendis (Batter) - Fitness: 0.4897350008517997
Kusal Perera (Batter) - Fitness: 0.5428491476312907
Pathum Nissanka (Batter) - Fitness: 0.4701019709134072
Kamindu Mendis (Batting Allrounder) - Fitness: 0.6741497304184305
Charith Asalanka (Batting Allrounder) - Fitness: 0.3947672473464449
Dasun Shanaka (Batting Allrounder) - Fitness: 0.8139283576538353
Angelo Mathews (Bowling Allrounder Fast) - Fitness: 0.3354386586051277
Maheesh Theekshana (Bowler Spin) - Fitness: 0.7146970766022269
Binura Fernando (Bowler Fast) - Fitness: 0.5438467710385408
Nuwan Thushara (Bowler Fast) - Fitness: 0.4874731686547863
```

Figure xxvi: GA Best Team Selection- Eng

```
Pathum Nissanka (Batter) - Fitness: 0.4241110183994889
Kusal Perera (Batter) - Fitness: 0.4412162875676148
Kusal Mendis (Batter) - Fitness: 0.3815111405255441
Charith Asalanka (Batting Allrounder) - Fitness: 0.3918888869847821
Kamindu Mendis (Batting Allrounder) - Fitness: 0.5841745934652716
Dasun Shanaka (Batting Allrounder) - Fitness: 0.691661258420519
Wanindu Hasaranga (Bowling Allrounder Spin) - Fitness: 0.9747421383647799
Maheesh Theekshana (Bowler Spin) - Fitness: 0.7523444199936387
Akila Dananjaya (Bowler Spin) - Fitness: 0.1562262627336889
Binura Fernando (Bowler Fast) - Fitness: 0.3967980569118733
Nuwan Thushara (Bowler Fast) - Fitness: 0.4081187518234953
```

Figure xxv: GA Best Team Selection- Ind

```
Kusal Perera (Batter) - Fitness: 0.542154552741509
Pathum Nissanka (Batter) - Fitness: 0.5058889413243026
Kusal Mendis (Batter) - Fitness: 0.4849980705741051
Charith Asalanka (Batting Allrounder) - Fitness: 0.4183268140969133
Kamindu Mendis (Batting Allrounder) - Fitness: 0.7206371272736953
Dasun Shanaka (Batting Allrounder) - Fitness: 0.8137213632930832
Angelo Mathews (Bowling Allrounder Fast) - Fitness: 0.3491034183320724
Maheesh Theekshana (Bowler Spin) - Fitness: 0.7623873728377691
Binura Fernando (Bowler Fast) - Fitness: 0.5391603750081788
Nuwan Thushara (Bowler Fast) - Fitness: 0.5223708464654543
```

Figure xxiv: GA Best Team Selection- NZ

```
Pathum Nissanka (Batter) - Fitness: 0.4701019709134072
Kusal Mendis (Batter) - Fitness: 0.4897350008517997
Kusal Perera (Batter) - Fitness: 0.5428491476312907
Dasun Shanaka (Batting Allrounder) - Fitness: 0.8139283576538353
Kamindu Mendis (Batting Allrounder) - Fitness: 0.6741497304184305
Charith Asalanka (Batting Allrounder) - Fitness: 0.3947672473464449
Angelo Mathews (Bowling Allrounder Fast) - Fitness: 0.3354386586051277
Maheesh Theekshana (Bowler Spin) - Fitness: 0.7146970766022269
Nuwan Thushara (Bowler Fast) - Fitness: 0.4874731686547863
Binura Fernando (Bowler Fast) - Fitness: 0.5438467710385408
```

Figure xxiii: GA Best Team Selection- SA

```
Pathum Nissanka (Batter) - Fitness: 0.4220309046136473
Kusal Perera (Batter) - Fitness: 0.4453503348829982
Bhanuka Rajapaksa (Batter) - Fitness: 0.3192910864711938
Kamindu Mendis (Batting Allrounder) - Fitness: 0.5693385592806777
Dasun Shanaka (Batting Allrounder) - Fitness: 0.699390074821455
Charith Asalanka (Batting Allrounder) - Fitness: 0.3774643802652937
Wanindu Hasaranga (Bowling Allrounder Spin) - Fitness: 0.9740251572327044
Chamika Karunaratne (Bowling Allrounder Fast) - Fitness: 0.4406067779803327
Maheesh Theekshana (Bowler Spin) - Fitness: 0.7721143457582287
Binura Fernando (Bowler Fast) - Fitness: 0.4111625526916087
Nuwan Thushara (Bowler Fast) - Fitness: 0.4123841786151778
```

Figure xxii: GA Best Team Selection- SL

```
Kusal Mendis (Batter) - Fitness: 0.4467891414766015
Kusal Perera (Batter) - Fitness: 0.30534853359966
Pathum Nissanka (Batter) - Fitness: 0.3929200858377856
Kamindu Mendis (Batting Allrounder) - Fitness: 0.3644038742391876
Dasun Shanaka (Batting Allrounder) - Fitness: 0.50020746498705
Danushka Gunathilaka (Batting Allrounder) - Fitness: 0.2578315763667379
Wanindu Hasaranga (Bowling Allrounder Spin) - Fitness: 0.9756981132075472
Angelo Mathews (Bowling Allrounder Fast) - Fitness: 0.3578777161496647
Maheesh Theekshana (Bowler Spin) - Fitness: 0.5871364171818702
Binura Fernando (Bowler Fast) - Fitness: 0.2910422131950153
Nuwan Thushara (Bowler Fast) - Fitness: 0.3114036356376987
```

Figure xxi: GA Best Team Selection- WI

5.4 Venue Prediction Model

This model allows users to input their selected playing XI and determine the best-suited venues for their team. The model was trained using historical match data, with K-Nearest Neighbors (KNN) identified as the best-performing algorithm.

```
Model: K-Nearest Neighbors
Accuracy: 0.9437
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.98      0.97     36179
           1       0.98      0.99      0.99     17935
           2       0.83      0.83      0.83      5978
           3       0.90      0.80      0.85     11908

    accuracy                           0.94     72000
   macro avg       0.92      0.90      0.91     72000
weighted avg       0.94      0.94      0.94     72000

Confusion Matrix:
 [[35627    26    47   479]
 [   54 17825    31    25]
 [  156   245  4974   603]
```

Figure xxx: Best ML model Selection - VPM

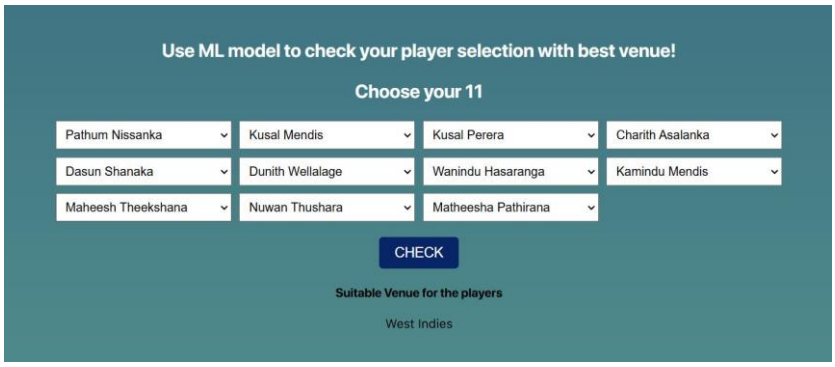


Figure xxxi: Web Interface- VPM

5.5 Fan Selection Probability Model

The fan-based selection model predicts the likelihood of a given playing XI being selected for the main squad. Data collected through Google Forms was used to create a favoritism score, which was analyzed using various machine learning models. XGBoost was found to be the most accurate model for predicting selection probabilities.

```
Model: Random Forest
MAE: 472.2437
MSE: 362600.2010
RMSE: 602.1629
R² Score: 0.2609
Accuracy: 67.0000

Model: Linear Regression
MAE: 537.2820
MSE: 480554.6042
RMSE: 693.2205
R² Score: 0.0205
Accuracy: 58.0000

Model: XGBoost
MAE: 412.7321
MSE: 269124.2001
RMSE: 518.7718
R² Score: 0.4515
Accuracy: 70.5000

Model: Decision Tree
MAE: 708.7100
MSE: 814688.7400
RMSE: 902.6011
R² Score: -0.6605
Accuracy: 48.5000
```

Figure xxxii: Best ML model Selection- FSPM

5.6 RAG AI Chatbot

An AI-powered Retrieval-Augmented Generation (RAG) chatbot was developed to provide answers related to Sri Lankan tournament cricket, school cricket, and domestic cricket. Unlike conventional chatbots, this model was trained on non-public cricket datasets, enabling it to answer questions that are not typically accessible via standard internet sources.

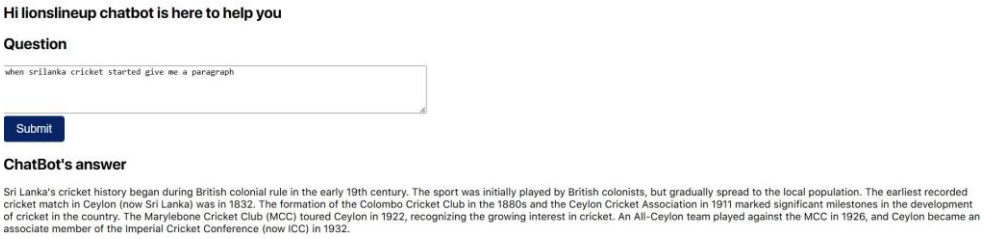


Figure xxxiii: Web Interface- RAG AI Chatbot

6 Conclusion and Future work

6.1 Conclusion

The primary objective of this study was to develop a data-driven approach for optimizing the selection of the Sri Lankan T20I cricket team for 2025. By integrating data from multiple sources including international match statistics from ESPN, LPL performance data from Papare, and fan opinions gathered through Google Forms this research successfully established an objective and systematic selection framework.

The methodology presented in this study encompasses several key components. First, cricket data was scraped and processed from diverse sources to ensure comprehensive player evaluation. Performance scores were then computed using a combination of statistical measures and machine learning-based weighting techniques to quantify player contributions accurately. To determine the optimal playing XI, a genetic algorithm was implemented, leveraging evolutionary principles to refine squad selection iteratively. Additionally, machine learning models were utilized to predict the best team composition for specific venues, further enhancing the adaptability of the selection process. Fan sentiment analysis was also incorporated to assess the likelihood of a player's selection based on public opinion. To make these insights accessible, a web-based application was developed, providing an interactive platform for users to explore the squad selection process.

The results demonstrate that this data-driven model significantly enhances squad selection by introducing a structured and evidence-based approach, reducing bias, and improving decision-making efficiency. The integration of multiple data sources ensures a holistic assessment of player performance, making the selection process more robust and transparent.

Despite these achievements, the study does have certain limitations. Data collection constraints, challenges in integrating different data formats, and the dynamic nature of cricket statistics pose ongoing challenges. However, the framework established in this research provides a strong foundation for further refinements and future advancements in cricket team selection methodologies.

6.2 Future Works

To enhance the system's efficiency and usability, future improvements should focus on automating the data collection and processing pipeline while refining user interactions. Several key advancements are proposed to achieve this goal.

First, Automated Data Retrieval will be implemented to scrape and update data from ESPN, Papare, and fan votes in real time, eliminating the need for manual intervention. This will ensure that the selection process is always based on the latest available data. Complementing this, Real-time Data Integration will involve developing a centralized database that dynamically updates with the latest cricket statistics, enabling a seamless flow of information for team selection.

To further refine the selection process, Enhanced Machine Learning Models will be developed by incorporating additional features such as player fitness, weather conditions, and opposition strengths. These factors will provide a more comprehensive analysis of match conditions and player readiness. Additionally, a User-Friendly Dashboard will be designed to offer interactive data visualizations, allowing users to explore various team compositions and simulate different scenarios with ease.

Another critical enhancement will be the introduction of an AI-Based Selection Assistant, an intelligent chatbot that will help users understand selection criteria, provide insights into squad composition, and answer cricket-related queries.

By integrating these advancements, the project can evolve into a fully automated, data-driven cricket team selection system, ensuring greater accuracy, efficiency, and transparency in decision-making.

7 References

[1] M. B. Malek, R. H. Badhan, M. I. Shesir and N. H. Fakir, Squad Selection for Cricket Team Using Machine Learning Algorithms, Dhaka, Bangladesh: Bachelor's thesis, Dept. of Computer Science and Engineering, BRAC University, 2018.

[2] V. D. Punjabi, R. B. Chaudhari, D. S. Pal, K. V. Nhavi, N. A. Shimpi and H. H. Joshi, A Survey on Team Selection in Game of Cricket Using Machine Learning, India: R.C. Patel Institute of Technology, 2023.

[3] I. Kumarasiria and S. Perera, Optimal One Day International Cricket Team Selection by Genetic Algorithm, International Journal of Sciences: Basic and Applied Research (IJSBAR), 2023.

[4] M. Robel, M. A. R. Khan, H. Kamrul, I. Ahammad and M. M. Alam, Cricket Players Selection for National Team and Franchise League using Machine Learning Algorithms, Department of Information and Communication Engineering, Noakhali Science and Technology University, Bangladesh, 2023.

[5] M. Gerber and G. D. Sharp, Selecting a Limited Overs Cricket Squad Using an Integer Programming Model, vol. 28, South African Journal for Research in Sport, Physical Education and Recreation, 2006, pp. 81-90.

[6] A. B. C. Jayaprada and K. P. Ihalapathirana, Player Fitness and Team Squad Selection using Intelligence System in Sri Lanka, vol. 183, Colombo: International Journal of Computer Applications, 2021.

8.Appendix

Player Role Selection according to Venue

```
import pandas as pd
import matplotlib.pyplot as plt

# Load the Excel file
file_path = '/content/All matches.xlsx'  # Replace with your actual file
path
excel_data = pd.ExcelFile(file_path)

# Function to generate combination counts for a given sheet
def generate_combination_counts(sheet_name):
    # Load the sheet
    df = pd.read_excel(file_path, sheet_name=sheet_name)

    # Create the combination dataframe
    combination_df = df[['Team', 'Bowling Allrounder Spin', 'Bowling
Allrounder Fast', 'Bowler Spin', 'Bowler Fast']]

    # Concatenate the values of the four categories to form the
"combination"
    combination_df['Combination'] = (combination_df['Bowling Allrounder
Spin'].astype(str) + '-' +
                                     combination_df['Bowling Allrounder
Fast'].astype(str) + '-' +
                                     combination_df['Bowler
Spin'].astype(str) + '-' +
                                     combination_df['Bowler
Fast'].astype(str))

    # Count the occurrences of each combination
    combination_counts = combination_df.groupby(['Team',
'Combination']).size().reset_index(name='Count')

    # Group by combination and sum the counts
    combination_counts_grouped =
combination_counts.groupby('Combination')['Count'].sum().reset_index()
```

```
    # Sort the combinations by count
    combination_counts_grouped =
combination_counts_grouped.sort_values('Count', ascending=False)

    return combination_counts_grouped

# Process each sheet: SENA, Srilanka, West Indies, India
combination_counts_sena = generate_combination_counts('SENA')
combination_counts_srilanka = generate_combination_counts('Srilanka')
combination_counts_west_indies = generate_combination_counts('West indies')
combination_counts_india = generate_combination_counts('India')

# Display results for each sheet
print("Combination Counts - SENA")
print(combination_counts_sena)

print("Combination Counts - Srilanka")
print(combination_counts_srilanka)

print("Combination Counts - West Indies")
print(combination_counts_west_indies)

print("Combination Counts - India")
print(combination_counts_india)

# Optionally, plot the bar charts for each sheet
def plot_combination_counts(combination_counts, title):
    plt.figure(figsize=(12, 7))
    plt.bar(combination_counts['Combination'], combination_counts['Count'],
color='skyblue')
    plt.xlabel('Bowling Combination (Bowling Allrounder Spin - Bowling
Allrounder Fast - Bowler Spin - Bowler Fast)')
    plt.ylabel('Count of Teams Using Combination')
    plt.title(title)
    plt.xticks(rotation=90)  # Rotate x-axis labels to make them readable
    plt.tight_layout()
    plt.show()
```

```
# Plot the combinations for each sheet
plot_combination_counts(combination_counts_sena, 'Distribution of Bowling
Combinations in SENA')
plot_combination_counts(combination_counts_srilanka, 'Distribution of
Bowling Combinations in Srilanka')
plot_combination_counts(combination_counts_west_indies, 'Distribution of
Bowling Combinations in West Indies')
plot_combination_counts(combination_counts_india, 'Distribution of Bowling
Combinations in India')

# Compare the best combinations across all sheets
def compare_best_combinations(*combination_counts_list):
    combined_df = pd.concat(combination_counts_list, ignore_index=True)
    best_combinations =
combined_df.groupby('Combination')['Count'].sum().reset_index()
    best_combinations = best_combinations.sort_values('Count',
ascending=False)
    return best_combinations

best_combinations = compare_best_combinations(combination_counts_sena,
combination_counts_srilanka, combination_counts_west_indies,
combination_counts_india)

# Display the best combinations
print("Best Bowling Combinations Across All Sheets")
print(best_combinations)
```

Performance Scoring System Implementation

IBaC – Weight Values

```
#importing libraries
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
```

```
from sklearn.linear_model import Lasso
from sklearn.inspection import permutation_importance
from sklearn.preprocessing import StandardScaler
data = pd.read_excel('/content/Allteamplayers - Copy - Copy.xlsx')
# Check the current column names
print("Column names (features):", data.columns)


# Display the DataFrame to confirm the changes
print(data.head())
X = data.drop("Runs",axis=1)    #Feature Matrix
y = data["Runs"]
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.3,
    random_state=0)

X_train.shape, X_test.shape
from sklearn.feature_selection import mutual_info_classif
# determine the mutual information
mutual_info = mutual_info_classif(X_train, y_train)
mutual_info
mutual_info = pd.Series(mutual_info)
mutual_info.index = X_train.columns
mutual_info.sort_values(ascending=False)
from sklearn.feature_selection import mutual_info_regression



# determine the mutual information
mutual_info1 = mutual_info_regression(X_train, y_train)
mutual_info1
mutual_info1 = pd.Series(mutual_info1)
mutual_info1.index = X.columns
mutual_info1.sort_values(ascending=False)
rf_model = RandomForestRegressor(random_state=0)
rf_model.fit(X_train, y_train)
rf_importances = pd.Series(rf_model.feature_importances_,
index=X.columns).sort_values(ascending=False)
xgb_model = XGBRegressor(n_estimators=10, max_depth=3, random_state=0,
use_label_encoder=False, eval_metric='rmse')
xgb_model.fit(X_train, y_train)
```

```
xgb_importances_scaled = pd.Series(xgb_model.feature_importances_,
index=X_train.columns).sort_values(ascending=False)
lasso_model = Lasso(alpha=0.1, random_state=0)
lasso_model.fit(X_train, y_train)
lasso_importances = pd.Series(lasso_model.coef_,
index=X_train.columns).sort_values(ascending=False)
permutation_result = permutation_importance(rf_model, X_test, y_test,
n_repeats=30, random_state=0)
permutation_importances = pd.Series(permutation_result.importances_mean,
index=X_train.columns).sort_values(ascending=False)


feature_importances_df = pd.DataFrame({
    "Random Forest": rf_importances,
    "XGBoost": xgb_importances,
    "Lasso Regression": lasso_importances,
    "Permutation Importance": permutation_importances
})
print(
    feature_importances_df.head(10)
)
```

IBaC – Performance Score

```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
data = pd.read_excel('/content/player wise Batting.xlsx')

# Check the current column names
print("Column names:", data.columns)

# Display the DataFrame to confirm the changes
print(data.head())
weights = {
    'Inns': 0.498829,
    '50': 0.047283,
```

```
    'Ave': 0.440953,
    '0': 0.000611,
    'SR': 0.011488,
    '100': 0.000835
}
data['Performance Score'] = (
    (weights['Inns'] * data['Inns']) +
    (weights['50'] * data['50s']) +
    (weights['Ave'] * data['Average']) +
    (weights['SR'] * data['Strike Rate']) +
    (weights['100'] * data['100s']) -
    (weights['0'] * data['0s'])
)
data.head()
```

Venue Prediction Model

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, accuracy_score
from sklearn.metrics import confusion_matrix
# Load the dataset

file_path = '/content/Model Data venue.xlsx'  # Update with your file path
df = pd.read_excel(file_path)

# Preprocessing
le = LabelEncoder()
df['Venue_Encoded'] = le.fit_transform(df['Venue'])
```

```
# Features and target
X = df[['Venue_Encoded', 'Fitness_Score']]
y = df['TeamID']

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Define models to evaluate
models = {
    "Logistic Regression": LogisticRegression(random_state=42,
max_iter=1000),
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "Random Forest": RandomForestClassifier(random_state=42),
    "SVM": SVC(random_state=42, probability=True),
    "KNN": KNeighborsClassifier()
}


# Train and evaluate models
results = {}
for model_name, model in models.items():
    # Train the model
    model.fit(X_train, y_train)
    # Make predictions
    y_pred = model.predict(X_test)
    # Evaluate the model
    accuracy = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred, zero_division=0,
output_dict=True)
    results[model_name] = {
        "Model": model,
        "Accuracy": accuracy,
        "Classification Report": report,
        "Confusion Matrix": confusion_matrix(y_test, y_pred)
    }

# Display the results
for model_name, result in results.items():
    print(f"Model: {model_name}")
    print(f"Accuracy: {result['Accuracy']:.4f}")
    print("Classification Report:")
    print(pd.DataFrame(result['Classification Report']).transpose())
    print("Confusion Matrix:")
```

```
    print(result['Confusion Matrix'])
    print("-" * 50)

# Identify the best model based on accuracy
best_model_name = max(results, key=lambda x: results[x]['Accuracy'])
best_model = results[best_model_name]['Model']
print(f"\nThe best model is: {best_model_name} with Accuracy:
{results[best_model_name]['Accuracy']:.4f}")



# Example Prediction with Best Model
def predict_team_with_best_model(venue, fitness_score):
    if venue not in le.classes_:
        raise ValueError(f"Venue '{venue}' is not in the dataset. Valid
venues are: {list(le.classes_)}")
    venue_encoded = le.transform([venue])[0]
    input_data = np.array([[venue_encoded, fitness_score]])
    predicted_team_id = best_model.predict(input_data)[0]
    team_details = df[df['TeamID'] == predicted_team_id]['Team'].values[0]
    return predicted_team_id, team_details

# Example: Predict the best team for a given venue and fitness score using
the best model
sample_venue = df['Venue'].iloc[0]  # Use an existing venue from the
dataset
sample_fitness_score = 80  # Example fitness score
predicted_team_id, predicted_team_details =
predict_team_with_best_model(sample_venue, sample_fitness_score)
print(f"\nThe best team for {sample_venue} with fitness score
{sample_fitness_score} is:")
print(f"Team ID: {predicted_team_id}")
print(f"Team Players: {predicted_team_details}")

# Simplify labels: Group teams by fitness score ranges (e.g., 0-30, 31-60,
61-100)
df['Team_Group'] = pd.cut(df['Fitness_Score'], bins=[0, 2, 3, 5],
labels=['Low', 'Medium', 'High'])

# Update target variable
y = df['Team_Group']

# Split dataset
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Train models again
results_grouped = {}
for model_name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    results_grouped[model_name] = {
        "Accuracy": accuracy,
        "Classification Report": classification_report(y_test, y_pred,
zero_division=0),
        "Confusion Matrix": confusion_matrix(y_test, y_pred)
    }

# Display grouped results
for model_name, result in results_grouped.items():
    print(f"Model: {model_name}")
    print(f"Accuracy: {result['Accuracy']:.4f}")
    print(result["Classification Report"])
    print(result["Confusion Matrix"])
    print("-" * 50)
```

Fan-Based Selection Model

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.linear_model import LinearRegression
from xgboost import XGBRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score
import pickle
```

```
df = pd.read_csv('/content/T20_Player_Responses__Constrained_.csv')
df.describe()
encoding_map = {
    'Akila Dananjaya': 0, 'Angelo Mathews': 1, 'Ashen Bandara': 2, 'Asitha
Fernando': 3,
    'Avishka Fernando': 4, 'Bhanuka Rajapaksa': 5, 'Binura Fernando': 6,
'Chamika Karunaratne': 7,
    'Charith Asalanka': 8, 'Dasun Shanaka': 9, 'Dhananjaya Lakshan': 10,
'Dhananjaya de Silva': 11,
    'Dilshan Madushanka': 12, 'Dilshan Munaweera': 13, 'Dinesh Chandimal':
14, 'Dunith Wellalage': 15,
    'Dushmantha Chameera': 16, 'Duvindu Tillakaratne': 17, 'Janith
Liyanage': 18, 'Jeffrey Vandersay': 19,
    'Kamindu Mendis': 20, 'Kasun Rajitha': 21, 'Kusal Mendis': 22, 'Kusal
Perera': 23,
    'Lahiru Kumara': 24, 'Lahiru Madushanka': 25, 'Lahiru Udara': 26,
'Lakshan Sandakan': 27,
    'Lasith Croospulle': 28, 'Maheesh Theekshana': 29, 'Matheesha
Pathirana': 30, 'Minod Bhanuka': 31,
    'Nishan Madushka': 32, 'Nuwan Thushara': 33, 'Nuwanidu Fernando': 34,
'Oshada Fernando': 35,
    'Pathum Nissanka': 36, 'Pramod Madushan': 37, 'Ramesh Mendis': 38,
'Sadeera Samarawickrama': 39,
    'Sahan Arachchige': 40, 'Vijayakanth Viyaskanth': 41, 'Wanindu
Hasaranga': 42
}
# Apply the encoding map to each column
for column in df.columns:
    df[column] = df[column].map(encoding_map)

# Now the DataFrame `df` has all player names converted to numbers
print("Encoded DataFrame:")
print(df)
X = df.drop(columns=['favoritism_score'])  # Encoded player data
y = df['favoritism_score']  # Favoritism scores as target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
models = {
    'Random Forest': RandomForestRegressor(n_estimators=100,
random_state=42),
    'Linear Regression': LinearRegression(),
    'XGBoost': XGBRegressor(n_estimators=100, random_state=42),
    'Decision Tree': DecisionTreeRegressor(random_state=42)
}
```

```
threshold = 0.1


results = {}

for model_name, model in models.items():
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)


    mae = mean_absolute_error(y_test, predictions)
    mse = mean_squared_error(y_test, predictions)
    rmse = mse ** 0.5
    r2 = r2_score(y_test, predictions)


    relative_error = np.abs((predictions - y_test) / y_test)
    accuracy = np.mean(relative_error < threshold) * 100  #



Percentage accuracy


    results[model_name] = {
        'MAE': mae,
        'MSE': mse,
        'RMSE': rmse,
        'R² Score': r2,
        'Accuracy': accuracy
    }


for model_name, metrics in results.items():
    print(f"\nModel: {model_name}")
    for metric_name, metric_value in metrics.items():
        print(f"{metric_name}: {metric_value:.4f}")
xgboost_model = models['XGBoost']


with open('model_XGBoost.pkl', 'wb') as file:
    pickle.dump(xgboost_model, file)
```

```
print("XGBoost model saved as 'model_XGBoost.pkl'.")
```

Genetic Algorithm Implementation

```
import pandas as pd
import random
class Player:
    def __init__(self, PlayerID, Role, Player, Fitness_Score):
        self.PlayerID = PlayerID
        self.Role = Role
        self.Player = Player
        self.Fitness_Score = Fitness_Score

    def __repr__(self):
        return f"{self.Player} ({self.Role}) - Fitness:
{self.Fitness_Score}"
file_path = '/content/Fitness Score.xlsx'
xls = pd.ExcelFile(file_path)
teams = {sheet_name: xls.parse(sheet_name) for sheet_name in
xls.sheet_names}



def create_pool(df, role):
    return [
        Player(row["PlayerID"], row["Role"], row["Player"],
row["Fitness_Score"])
        for _, row in df.iterrows() if row["Role"] == role
    ]
def create_individual(team_name, composition):
    df = teams[team_name]
    selected_ids = set()
    team = []

    for role, count in composition.items():
        pool = create_pool(df, role)
        random.shuffle(pool)  # Shuffle to add randomness
```

```
        role_players = [p for p in pool if p.PlayerID not in
selected_ids][:count]
        team.extend(role_players)
        selected_ids.update(p.PlayerID for p in role_players)

    return team

def calculate_fitness(individual):
    return sum(player.Fitness_Score for player in individual)

def mutate(individual, team_name):
    df = teams[team_name]
    existing_ids = {player.PlayerID for player in individual}
    random_index = random.randint(0, len(individual) - 1)
    role = individual[random_index].Role

    pool = [p for p in create_pool(df, role) if p.PlayerID not in
existing_ids]

    if pool:  # Only replace if a valid player exists
        individual[random_index] = random.choice(pool)

    return individual



def crossover(parent1, parent2):
    child1, child2 = parent1[:], parent2[:]
    crossover_point = random.randint(1, len(parent1) - 1)

    # Swap up to the crossover point
    for i in range(crossover_point):
        child1[i], child2[i] = child2[i], child1[i]

    # Remove duplicates
    def remove_duplicates(child):
        seen_ids = set()
        unique_child = []
        for player in child:
            if player.PlayerID not in seen_ids:
                unique_child.append(player)
                seen_ids.add(player.PlayerID)
        return unique_child
```

```
    return remove_duplicates(child1), remove_duplicates(child2)




def genetic_algorithm(team_name, composition, population_size=50,
generations=100, mutation_rate=0.1, crossover_rate=0.8):
    population = [create_individual(team_name, composition) for _ in
range(population_size)]

    for generation in range(generations):
        fitness_scores = [calculate_fitness(individual) for individual in
population]
        population = [x for _, x in sorted(zip(fitness_scores, population),
key=lambda pair: pair[0], reverse=True)]

        new_population = population[:population_size // 2]

        while len(new_population) < population_size:
            parent1, parent2 = random.sample(population[:population_size //
2], 2)

            if random.random() < crossover_rate:
                child1, child2 = crossover(parent1, parent2)
            else:
                child1, child2 = parent1[:], parent2[:]

            if random.random() < mutation_rate:
                child1 = mutate(child1, team_name)
            if random.random() < mutation_rate:
                child2 = mutate(child2, team_name)

            new_population.extend([child1, child2])

        population = new_population[:population_size]
        print(f"Generation {generation + 1}: Best Fitness =
{calculate_fitness(population[0])}")

    return population[0]

team_compositions = {
    "New Zealand": {"Batter": 3, "Batting Allrounder": 3, "Bowling
Allrounder Fast": 1, "Bowler Spin": 1, "Bowler Fast": 2},
```

```
    "Australia": {"Batter": 3, "Batting Allrounder": 3, "Bowling Allrounder
Fast": 1, "Bowler Spin": 1, "Bowler Fast": 2},
    "England": {"Batter": 3, "Batting Allrounder": 3, "Bowling Allrounder
Fast": 1, "Bowler Spin": 1, "Bowler Fast": 2},
    "South Africa": {"Batter": 3, "Batting Allrounder": 3, "Bowling
Allrounder Fast": 1, "Bowler Spin": 1, "Bowler Fast": 2},
    "Sri Lanka": {"Batter": 3, "Batting Allrounder": 3, "Bowling Allrounder
Spin": 1, "Bowling Allrounder Fast": 1, "Bowler Spin": 1, "Bowler Fast":
2},
    "West Indies": {"Batter": 3, "Batting Allrounder": 3, "Bowling
Allrounder Spin": 1, "Bowling Allrounder Fast": 1, "Bowler Spin": 1,
"Bowler Fast": 2},
    "India": {"Batter": 3, "Batting Allrounder": 3, "Bowling Allrounder
Spin": 1, "Bowler Spin": 2, "Bowler Fast": 2},
    "Bangladesh": {"Batter": 3, "Batting Allrounder": 3, "Bowling
Allrounder Spin": 2, "Bowler Spin": 1, "Bowler Fast": 2}
}
for team, composition in team_compositions.items():
    print(f"\nBest Team for {team}:")
    best_team = genetic_algorithm(team, composition)
    for player in best_team:
        print(player)
```

RAG AI Chatbot

```
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel
import os
from contextlib import asynccontextmanager
from fastapi.middleware.cors import CORSMiddleware
from langchain_groq import ChatGroq
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.chains.combine_documents import create_stuff_documents_chain
from langchain_core.prompts import ChatPromptTemplate
from langchain.chains import create_retrieval_chain
from langchain_community.vectorstores import FAISS
from langchain_community.document_loaders import PyPDFDirectoryLoader
from langchain_google_genai import GoogleGenerativeAIEmbeddings
from dotenv import load_dotenv
import time

load_dotenv()
```

```
# Global variable for vector store
vectors = None

@asynccontextmanager
async def lifespan(app: FastAPI):
    # Load documents on startup
    global vectors
    embeddings = GoogleGenerativeAIEmbeddings(model="models/embedding-001")
    loader = PyPDFDirectoryLoader("./us_census")
    docs = loader.load()
    text_splitter =


RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=200)
    final_documents = text_splitter.split_documents(docs[:20])
    vectors = FAISS.from_documents(final_documents, embeddings)
    print("Vector Store DB is ready")
    yield
    # Clean up resources if needed
    vectors = None


# Initialize FastAPI with lifespan
app = FastAPI(lifespan=lifespan)

# Add CORS middleware configuration
app.add_middleware(
    CORSMiddleware,
    allow_origins=[
        "http://localhost:3000",
        "http://localhost:4000",
        "http://localhost:5173",
    ],  # Include localhost:3000
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)



# Load the GROQ and Google API Keys
groq_api_key = os.getenv("GROQ_API_KEY")
os.environ["GOOGLE_API_KEY"] = os.getenv("GOOGLE_API_KEY")
```

```
# Initialize the LLM and prompt template
llm = ChatGroq(groq_api_key=groq_api_key, model_name="Llama3-8b-8192")
prompt = ChatPromptTemplate.from_template(
    """

    Please provide the most accurate response based on the question.
    {context}
    Questions:{input}
    """
)

"""    <context>
    {context}
    <context>
    Questions:{input}"""


# Create vector embedding function
class QuestionRequest(BaseModel):
    question: str


@app.post("/ask-question")
async def ask_question(request: QuestionRequest):
    """Endpoint to answer questions based on the document data."""
    if not vectors:
        raise HTTPException(status_code=500, detail="Vector Store is not
initialized")

    document_chain = create_stuff_documents_chain(llm, prompt)
    retriever = vectors.as_retriever()
    retrieval_chain = create_retrieval_chain(retriever, document_chain)

    start = time.process_time()
    response = retrieval_chain.invoke({"input": request.question})

    response_time = time.process_time() - start

    if "answer" not in response:
        raise HTTPException(status_code=404, detail="Answer not found")

    return {
        "answer": response["answer"],
```

```
        "response_time": response_time,
        # "context": [doc.page_content for doc in response.get("context",
[])]
    } [1] [2]
```